BAYESIAN 3D MULTIPLE PEOPLE TRACKING USING MULTIPLE INDOOR CAMERAS AND MICROPHONES

A Thesis
Presented to
The Academic Faculty

by

Yeongseon Lee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2009

BAYESIAN 3D MULTIPLE PEOPLE TRACKING USING MULTIPLE INDOOR CAMERAS AND MICROPHONES

Approved by:

Professor Rusell M. Mersereau,
Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Georgia Vachtsevanos
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Biing Hwang (Fred) Juang
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Christopher E. Heil
School of Mathematics
Georgia Institute of Technology

Professor James H. McClellan
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: 14 April 2009

To my parents, husband, and my daughter,

who have supported for me with trust and patience

ACKNOWLEDGEMENTS

The time I have been in Georgia Tech for PhD research not only means studying a particular area deeper but also means learning how to live and work in the new culture and environment, which I was not familiar with. I have been struggling in this new and challenging environment, but I always knew I could manage it. Now I finally came to the place I can close one epoch of my life, and I feel deeply gratitude to people around me. They mostly encouraged me to go forward, but they sometimes, frankly, made myself shameful when I compared myself with them since they were the smartest and hard-working of all.

First, I like to thank to my adviser, Dr. Mersereau. He was totally different person I have ever known. He always encouraged me. He was always there whenever I needed him. However, I always feel sorry not to be a good student to him in some ways. I always feel regret I could not make a good relationship with him beyond the student and teacher relationship, partly because I was a really stubborn Korean, and mostly because I could not make research progress quickly. However, he has never given me up. I really appreciate it.

I also thank to Dr. McClellan, Dr. Juang, and Dr. Pappas in Northwestern University. It was not only because they were in my committee but also because they taught me how I approach my research (work) in my life by showing me their steadiness, eagerness and patience toward their research.

I also want to thank to all my colleagues, Volkan Cevher, Nazanin Rahnavard, Mubashir Alam, Toygar Akgun, Milind Borkar, Nicolas Gastaud, Ali Cafer Gurbuz, who already graduated and settled their life in some places, and Ted Wada, Salman

Aslam, Gregory Krudysz, Chengyuan Ma, Antonio Moreno-Daniel, Enrique Robledo-Arnuncio, Gaofeng Yue, and Kaustubh Kalgaonkar who are still with me at school. I learned a lot of things from them more than from literatures and professors. I hope I can keep the relationship with them wherever we work.

Most of all, I like to thank to my husband, who already finished the same process before me. I am sure I could not finish it without his support. I also appreciate my parents who concerned their second daughter's new challenge for PhD in U.S. Since they had never worried about me at least for studying, it was much harder for them to watch me during this period. At last, I like to thank to my daughter, my most beloved and precious one, I don't know how she will remember me during this time in the future, and I don't know how this period time affected her. However, I hope she will get some courage from me when she has to challenge her life with something she has never done before.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis represents Bayesian joint audio-visual tracking for the 3D locations of multiple people and a current speaker(s) in a real conference environment. To achieve this objective, it focuses on several different research interests, such as acoustic-feature detection, visual-feature detection, a tracking framework, data association, and sensor fusion. As acoustic-feature detection, time-delay-of-arrival (TDOA) estimation is used for the detection of multiple acoustic sources. Localization performance using TDOAs is also analyzed according to different configurations of microphones. As visual-feature detection, Viola-Jones face detection is used to initialize the locations of unknown multiple people. Then, motion detection using a corner feature, based on the results from the Viola-Jones face detection, is used to follow these non-rigid frontal faces/face profile/upper bodies in normal tracking mode. Simple point-to-line correspondences between multiple cameras using fundamental matrices are used to determine which features are more robust. As a method for data association and sensor fusion, Monte-Carlo JPDAF and a data association with IPPF (DA-IPPF) are implemented in the framework of particle filtering. The proposed algorithms and framework are applied to three different tracking scenarios of acoustic source tracking, visual source tracking, and joint acoustic-visual source tracking. Finally the implementation of this joint acoustic-visual tracking system using cameras and microphones is introduced in two parts of system implementation and real-time processing.

CHAPTER I

INTRODUCTION

1.1 Overview

Target tracking is an old research topic, especially in the military, which uses radar, sonar, or seismic sensors. However, as a result of the current popularity of the internet, high-speed networks, fast processors, and affordable sensing equipment, new target-tracking applications, which use visual and acoustic sensors, have appeared in commercial areas. The most representatives of these applications so far have been video conferences, visual surveillance, and scene analysis.

Video conferences, based on the high-speed internet, have inspired the research for more realistic, intelligent conference systems. Some research has implemented these systems using high-resolution images, larger screen displays, and low delays between both sides by developing better devices and networks. However, this is not our research interest here. Other research has concentrated on implementing an automatic camera steering system to track a "current speaker" to minimize the need for a human-operated camera by localizing the accurate position of the speaker. In this work, the current speaker refers to the person who is speaking at the current time and can be any of the people attending the conference. This idea of localizing the current speaker has been tried by using either acoustic features from microphone arrays or visual features from camera images.

Visual surveillance is also greatly needed these days both indoors, such as in offices, hospitals, and governmental buildings, and outdoors, such as in parking lots and airports. Visual surveillance systems have been implemented by using multiple cameras installed in different locations. Like video conference systems, they localize,

track, or recognize people by focusing on visual features such as faces, human bodies, and/or motion information. They also use different sensors such as microphones, infra-red badges, or lasers.

Another interesting application relevant to acoustic or visual tracking is scene analysis. The *Aware House* [19] project at Georgia Tech, for example, focuses on developing a model of a future home and aims to provide services for its residents, so they can maintain independence as they age. Providing this kind of assistance requires location detection and activity recognition capability. Another example of indoor scene analysis is *CHIL*, which, under the European Commission's Sixth Framework Programme [13], focuses on creating environments where computers serve people by helping them interact with each other. This requires details of the people's states, activities, and even intentions. Even though these two projects have different purposes, tracking subjects in their environments is required in both.

Since acoustic or visual tracking is fundamental to these applications just mentioned, this research focuses on detecting the locations of subjects. More specifically, it proposes to track the three-dimensional (3D) locations of people and the current speaker/speakers in a real conference environment in which the movements of people or the locations of objects are not restricted for tracking purposes. In a real conference environment, it is impossible to develop a perfect location tracker to cover all irregular movements or activities, but we aim to detect/track the 3D locations of multiple people and a current speaker(s) in time under non-stationary conditions with acceptable accuracy.

To achieve this objective, this research first considers the capabilities and weaknesses of sensors such as cameras and microphones. The advantage of camera images is that they provide various rich visual features, not only of the shapes of faces and bodies, but also of the color differences among faces, bodies, and background. They

also easily detect motion from consecutive images, which indicates primarily the existence of foreground objects in contrast to a static background. One drawback of using cameras, however, is that at least two cameras are required to detect the 3D locations of attendants. Another drawback of it is that the angle of a view in a camera is a limited value. Moreover, in the case of low-resolution images, the detection of a speaker is not an easy task because with such images, the specific target area, in this case the mouth of the speaker, is usually not clear. An environment with multiple people makes the detection of a current speaker even more difficult, because the current speaker can be blocked by other people and not be seen in camera images. Therefore, one might logically conclude that tracking a current speaker would be easier if the approach focused on tracking by using speech. However, speech signals are intermittent, and acoustic clutter, such as laughing, clapping, reverberation, or other noise, disturbs accurate localization. Therefore, this research proposes to use both cameras and microphones even though the computational complexity is increased. The goal of this research is to demonstrate that using cameras and microphones together can compensate for the most serious drawbacks of each, resulting in more robust tracking.

This research also evaluates the efficiency and accuracy of a number of algorithms for acoustic and visual-feature detection. Even though there has been a great deal of research on both kinds of detection algorithms, it is not easy to design appropriate acoustic and visual feature-detection algorithms in a real conference environment as described above. The goal here for acoustic-feature detection is to find appropriate acoustic features for the detection of multiple speakers using a small number of microphones. The goal for visual-feature detection is, while using low-resolution images, to find appropriate visual features for the detection of multiple people who are moving in an irregular manner.

To combine different measurements from cameras and microphones, this research requires sensor fusion. To accomplish this, the research proposes using a Bayesian

framework. This framework treats measurements and final locations as random variables and combines them in a probabilistic way using a joint probability density. By considering non-linear relations between measurements and final 3D locations, this research specifically uses a non-linear Bayesian method called a particle filter.

In target tracking, ambiguity between measurements and objects is inevitable, which is a common issue of data association. This ambiguity can also occur as a result of losing measurements or obtaining incorrect measurements. This research solves these data association problems within the Bayesian framework using a particle implementation.

In summary, this research proposes joint audio-visual tracking for the 3D locations of multiple people and a current speaker in a real conference environment. To achieve this objective, it focuses on several different research interests, such as acoustic-feature detection, visual-feature detection, a non-linear Bayesian framework, data association, and sensor fusion.

1.2 Outline of the Thesis

Chapter 2 begins with research background to provide a big picture of the entire research. The background is divided into five parts: acoustic-feature detection, visual-feature detection, a conventional particle filter, data association, and sensor fusion. Each part briefly introduces the most representative algorithms and discusses their advantages and weaknesses.

Chapter 3 introduces acoustic and visual-feature detection. Acoustic-feature detection is based on time-delay-of-arrival (TDOA) estimation. Localization using TDOAs will be analyzed according to different configurations of microphones. Visual-feature detection uses Viola-Jones face detection as an initialization method. Then, a corner feature, based on the results from the Viola-Jones face detection, is used for motion detection for a specific object. Simple point-to-line correspondences between

multiple cameras using fundamental matrices are used to determine which features are more robust.

Chapter 4 introduces approaches for both data association and sensor fusion. This chapter first introduces a data association parameter, a joint probabilistic data association filter (JPDAF). Then, it develops and evaluates two algorithms of Monte-Carlo JPDAF and a data association with IPPF (DA-IPPF), which are implemented in the framework of particle filtering. It then goes on to propose an initialization method to find the number and initial locations of the objects by using multiple data association hypotheses.

Chapter 5 shows multiple target tracking using sensors of a single type, such as acoustic tracking using multiple microphones and visual tracking using multiple cameras by using both the features detection of Chapter 3 and the framework of Chapter 4. Simple data models are proposed for the particle filter. Then, according to different scenarios of data association methods, measurement noise, falsely detected data, and missing data, simulation demonstrates the results of the initialization and multiple target tracking.

Chapter 6 shows multiple people tracking and speaker activity detection using both microphones and cameras. The simulation results reveal performance improvement as a result of using joint sensors compared to using only microphones or cameras.

Chapter 7 addresses the real-time implementation of this joint acoustic-visual tracking system using a PC, four cameras, and six microphones. This chapter is divided into two parts consisting of system implementation and real-time processing. The part dealing with system implementation introduces the components of the real system. The part that explains real-time processing focuses on reducing both processing time and computational complexity.

CHAPTER II

BACKGROUND

2.1 Overview

A system for Bayesian multiple-target tracking using multiple cameras and microphones can be represented as having three different blocks: sensors, source-feature detection, and non-linear Bayesian target tracking, as shown in Figure 2.1. While

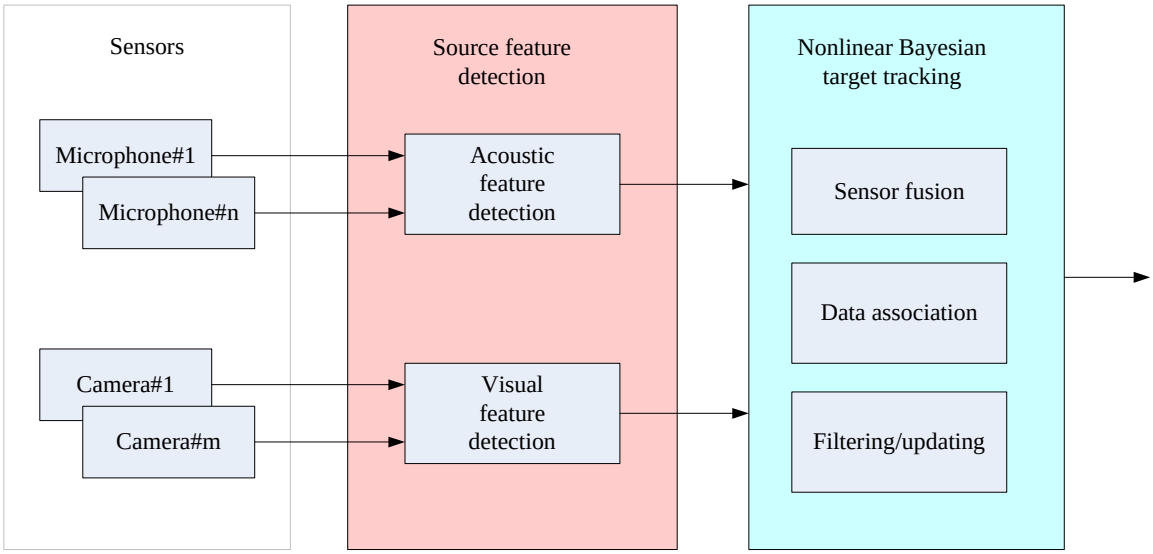


Figure 2.1: The block diagram of a Bayesian multiple-target tracking system using multiple cameras and microphones.

the design of the sensors is outside the scope of this research, the latter two blocks, source feature detection and non-linear Bayesian target tracking, are the major interests. Source-feature detection is divided into acoustic-feature detection, which uses microphones, and visual-feature detection, which uses multiple cameras. Non-linear Bayesian target tracking is divided into three components: updating/filtering, data association, and sensor fusion. The updating/filtering uses a state-space approach to model a dynamic system, the data association clarifies the relationship between

unlabeled measurements and targets, and the sensor fusion merges the different measurements from the cameras and microphones. While acoustic and visual-feature detection operate independently, the three sub-components of the non-linear Bayesian target tracking are integrated into the single unified framework of particle filtering.

2.2 Acoustic Feature Detection

Acoustic-feature detection is an active research area in acoustics and sonar. It estimates the location features of a single source or multiple acoustic sources using multiple microphones at different locations. These location features are a 2D/3D position, a range, or a direction-of-arrival (DOA). The common algorithms applied to location feature estimation are time-delay estimation methods (TDE) [31–33, 62, 67] and beamforming techniques [3, 10, 11, 76].

The TDE methods, which are sometimes called two-step TDE methods, are used to estimate the relative time delay between pairs of microphones. The first step estimates the time delay of arrival (TDOA). The most representative methods for TDOA estimation are a generalized cross-correlation (GCC), a phase transform (PHAT), and an adaptive eigenvalue decomposition algorithm (AEDA) [31]. The second step determines a final source location by using the intersection of a set of hyperbolic curves derived from the estimated multiple TDOAs. The most representative methods for source localization are an iterative maximum likelihood (ML) method and a spherical-interpolation method [33, 62]. These TDE methods are commonly used because they are quite fast, robust, and easy to implement in an ideal free field. However, they are known to be limited in that they can detect only a single source.

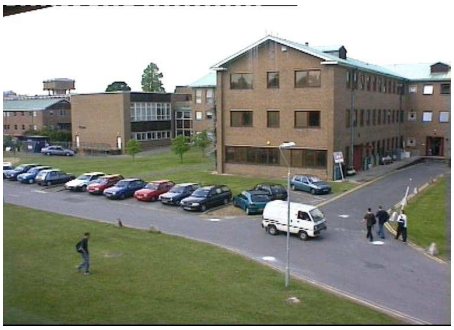
In addition to the TDE method, the DOAs of a single source or multiple sources can be estimated by beamforming methods [38, 40]. Some of the most well known of these methods are a delay-and-sum beamformer, a minimum variance distortion beamformer (MVDR), multiple signal classification (MUSIC) [64], and root MUSIC.

The capability of beamformers to carry out multiple source detection has great potential for multiple object tracking. However, because beamforming searches all possible regions, one of its drawback of the beamforming is that, compared to TDE methods, it usually requires more processing time. Another drawback is that since beamformers are built on pre-defined signal models (such as a wide or a narrow band signal and a spherical-wave or a plane-wave propagation model), the performance of beamformers is degraded if the models are mismatched in a real environment.

Even though both TDE and beamforming methods work well in an ideal free field, their performance degrades in an environment with reverberation or severe noise. Rather than attacking reverberation directly, one way to solve this problem is to apply non-linear Bayesian filtering [8,17,53,67,71,76]. Non-linear Bayesian filtering utilizes the fact that the detected measurements caused by reverberation do not have temporal consistency while the detected measurements from the true sources do have temporal consistency.

2.3 Visual Feature Detection

As mentioned before, visual surveillance [5, 65, 66] and conference systems are two dominant applications for tracking people [23, 52]. While outdoor surveillance systems focus on detecting whole bodies of moving people with faraway cameras, conference systems and indoor surveillance systems focus on detecting only upper bodies or faces/heads using close-range cameras. This is demonstrated in Figure 2.2 and Figure 2.3. In Figure 2.2, the sizes of the people in an outdoor surveillance system are small, and the motion of the people is the main feature used to distinguish them from the background [37]. Compared to the sizes of the people in Figure 2.2, the sizes of the people in the conference system in Figure 2.3 are relatively large and do not have distinct, directional motion. Therefore, in conference systems, a face feature is most commonly used.



(a) An image from camera 1.

(b) An image from camera 2

Figure 2.2: Sample surveillance images using two cameras from PETS2001.

(a) An image from camera 1

(b) An image from camera 2

Figure 2.3: Sample images captured from our conference system with two cameras.

Face detection using images, which focuses primarily on the detection of frontal faces, has been extensively researched in the past 30 years, but, surprisingly, this research area is still challenging. Yang [74] classified face-detection research prior to 2000 into four categories: knowledge-based, feature-based, template-based, and appearance-based. Among the four categories, methods in feature-based and appearance-based categories are more popular than the others.

In addition to Yang's classification, face detection based on Freund's Adaboost [18] was proposed by Viola [69] in 2001, which made a great impact on face-detection research. It led face-detection research in the direction of developing algorithms for

real-time implementation [48]. In this research, the algorithm proposed by Viola and Jones [69] will be called a Viola-Jones detector. Three main components of the Viola-Jones detector are its use of an integral image to reduce computational time, an Adaboost algorithm with weak classifiers to train face and non-face models, and a cascade search. Each of these components has resulted in additional research that has furthered the field [41,46,48]. The fact that OpenCV [35], which is an open computer vision library from INTEL, added the Viola-Jones detector to its library has provided easy access to this algorithm. To overcome its frontal face detection limitation, the Viola-Jones detector added rotated features of frontal faces so that it could detect rotated frontal faces [41, 46, 48]. The Viola-Jones detector has also been extended to the non-frontal face detection of face profiles, upper bodies, or whole bodies [29, 70].

While the methods introduced so far, called frame-based approaches here, were developed based on a single image, some detection algorithms have been developed for using video sequences such as a blob detector [36] and a mean-shift algorithm [15]. One advantage of these video-based approaches is that they can incorporate any frame-based algorithms. Another advantage is that they exploit temporal correlation between consecutive frames. Therefore, video-based approaches usually assume that objects have motion. This means that they can extract foreground objects using frame differences or background extraction algorithms [37, 63]. Another advantage of video-based approaches is that they can reduce computational time by using temporal correlations. Since the current position for objects can be assumed from the previous position of the objects using temporal correlation, objects can be found without the need to search whole images. However, one weakness of video-based approaches is that the success of object detection in video usually requires accurate initialization.

2.4 Non-linear Bayesian Tracking: Particle Filter

While the two previous sections introduced several algorithms for extracting appropriate acoustic and visual features for targets from cameras and microphones, this section describes how to build a tracking system with the measurements from the feature extraction and known information about the tracking system. A Bayesian framework, which is a probabilistic framework [4, 12], treats any kind of measurements and interests about targets, 3D locations here (called a state space), as random variables and attempts to construct a joint probability density of the state space. All known information about the tracking system, called the prior information, is also described with a probability and reflected to the tracking system. For simplicity of explanation, this section introduces mathematical symbols and terminology. A variable to be estimated from tracking is called a state space and is denoted by $\mathbf{x}$. $\mathbf{z}$, measurements, indicates extracted features from sensors. t indicates the current time, and $0 : t$ indicates time up to time t beginning at $t = 0$. A matrix is described with a bold, capital letter, a vector with a bold, lowercase letter, and a scalar with a lowercase, regular letter. $\mathbf{x}_t$ is a state-space vector at time t, $\mathbf{z}_t$ is a measurement vector at time t as $\mathbf{z}_t = [z_{1,t}, \ldots, z_{n,t}]^T$, and $\mathbf{Z}_t$ is a measurement matrix up to time t as $\mathbf{Z}_t = [\mathbf{z}_1, \ldots, \mathbf{z}_t]$. Instead of $\mathbf{Z}_t$, $\mathbf{z}_{1:t}$ is sometimes used to mean the same when it needs to show time clearly.

Bayesian tracking is built on a state-space model, a hidden Markov model, and Bayesian estimation. A system in a Bayesian framework is described by using a state-space model with two parts: a state update model as Equation (1) and a measurement model as Equation (2).

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_t) \tag{1}$$

$$\mathbf{z}_t = h_t(\mathbf{x}_t, \mathbf{n}_t) \tag{2}$$

$f_t(.)$ and $h_t(.)$ are linear or non-linear functions, which describe the relationship between $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ and between $\mathbf{x}_t$ and $\mathbf{z}_t$. These two functions are used to estimate probabilities computed from a prior probability and a data likelihood.

A hidden Markov model in the Bayesian tracking, specifically a first-order Markov model, simplifies the relationship among the current and past state spaces and measurements. When the first-order Markov model is applied to the state-space model, the current state space depends only on the previous state. As a result, the prior probability simplifies to $p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$.

The Bayesian estimation in Bayesian tracker estimates a certain analytic function by using its posterior probability as

$$E(f(\mathbf{x})) = \int f(\mathbf{x})p(\mathbf{x}|\mathbf{z})d\mathbf{x} \tag{3}$$

where $f(\mathbf{x})$ is an analytic function, and $p(\mathbf{x}|\mathbf{z})$ is the posterior probability of $\mathbf{x}$. If the state is the quantity tracked, as in this research, the Bayesian estimation of $\mathbf{x}$ is just the mean value of the state.

$$E(\mathbf{x}) = \int \mathbf{x}p(\mathbf{x}|\mathbf{z})d\mathbf{x} \tag{4}$$

A major problem in Bayesian tracking is how to estimate the posterior probability of the state at each time. Since this posterior probability cannot be estimated directly, Bayes' theorem is used as shown below.

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{Z}_t) &= \frac{p(\mathbf{Z}_t|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{Z}_t)} \\ &= \frac{p(\mathbf{z}_t, \mathbf{Z}_{t-1}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t, \mathbf{Z}_{t-1})} \\ &= \frac{p(\mathbf{z}_t|\mathbf{Z}_{t-1}, \mathbf{x}_t)p(\mathbf{Z}_{t-1}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})p(\mathbf{Z}_{t-1})} \\ &= \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1})p(\mathbf{Z}_{t-1})p(\mathbf{x}_t)}{p(\mathbf{x}_t)p(\mathbf{z}_t|\mathbf{Z}_{t-1})p(\mathbf{Z}_{t-1})} \\ &= \frac{p(\mathbf{x}_t|\mathbf{Z}_{t-1})p(\mathbf{z}_t|\mathbf{x}_t)}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})} \end{aligned} \tag{5}$$

The numerator in Equation (5) is calculated with a filtering process using the priori probability and a previously estimated state, as in Equation (6), and an updating process using a data likelihood, as in Equation (7). The denominator, called a predictive measurement, is simply the integral of the numerator with respect to $\mathbf{x}_t$ as in Equation (8).

$$\text{Filtering}: p(\mathbf{x}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1})d\mathbf{x}_{t-1} \tag{6}$$

$$\text{Updating}: p(\mathbf{x}_t|\mathbf{Z}_t) = p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1}) \tag{7}$$

$$\text{Predictive measurement}: p(\mathbf{z}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1})d\mathbf{x}_t \tag{8}$$

The most famous and successful method of Bayesian tracking is a Kalman filter [12]. However, a Kalman filter is based on two assumptions: a linear system and a Gaussian posterior probability. In reality, many tracking systems have non-linear state update and/or data likelihood models, so a Kalman filter cannot be directly applied to non-linear systems.

To adapt the Kalman filter to a non-linear system, an extended Kalman filter [7,12] and an unscented Kalman filter [12,50] were developed and successfully applied. Since an extended Kalman filter is the local linearization of a non-linear function using a first-order Taylor expansion, it is still limited in its ability to describe the function satisfactorily. An unscented Kalman filter samples several points around a mean value and then propagates these samples into a non-linear function. While an extended Kalman filter keeps only the first-order statistics, an unscented Kalman filter can keep the first-order, second-order, and even higher statistics. Therefore, it is known to track a non-linear function much better than an extended Kalman filter.

A totally different approach that describes a probability with weighted discrete samples was developed [4, 12, 17, 76]. Using this approach, Bayesian tracking tries to implement a non-linear posterior probability and a data likelihood. This attempt to

describe a posterior probability as weighted discrete samples originated from Monte-Carlo approximation methods [12], which, in turn, were developed to estimate difficult integral functions. When a Monte-Carlo method is recursively applied to estimate a posterior probability in time, this method is called a sequential Monte-Carlo method. This is also called a sequential Monte-Carlo method or a particle filter. This research uses a particle filter as its basic framework for non-linear Bayesian tracking.

2.4.1 Sequential Importance Sampling

In a particle filter, a posterior distribution of $\mathbf{x}$ at time t is described with N samples as

$$p(\mathbf{x}_t|\mathbf{Z}_t) \approx \sum_{n=1}^{N} w_t^{(n)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(n)}) \tag{9}$$

where $\mathbf{x}_t^{(n)}$ and $w_t^{(n)}$ denote discrete samples and their corresponding associated weights. Then, a major problem in the implementation of the particle filter becomes more specific: how to generate these samples and calculate their weights.

The samples and weights are generated using importance sampling [12]. In importance sampling, samples are generated by a probability called an importance density or a proposal function, $q(\mathbf{x})$. This importance density should be easily implemented and cover the support region of the true posterior probability. To calculate the weights, an analytic probability $\pi(\mathbf{x})$, which has the property that $\pi(\mathbf{x}) \propto p(\mathbf{x})$, should be proposed. Here, $p(\mathbf{x})$ is the unknown, true probability of $\mathbf{x}$. Then, the weights, $w_t^{(n)}$ in Equation (9), are approximated by the ratio of $\pi(\mathbf{x})$ to the importance density, $q(\mathbf{x})$, as

$$w_t^{(n)} \propto \frac{\pi(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}{q(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}. \tag{10}$$

That $\mathbf{x}_t^{(n)}$ and $w_t^{(n)}$ are approximations of the true probability of $\mathbf{x}$ can be shown by using Bayesian estimation of Equation (4). Equation (4) is approximated as in

Equation (11) using a Monte-Carlo approximation

$$\hat{E}(\{\mathbf{x}_t) = \sum_{n=1}^{N} \mathbf{x}_t^{(n)} p(\mathbf{x}_t^{(n)}|Z)\delta(\mathbf{x}_t - \mathbf{x}_t^{(n)}) \tag{11}$$

where $\mathbf{x}_t^{(n)}$ is assumed to be generated by its posterior $p(\mathbf{x}_t|\mathbf{z}_t)$. When the numerator and denominator in Equation (4) are multiplied by $q(\mathbf{x}_t|\mathbf{z}_t)$, it is described as

$$E(\mathbf{x}_t) = \int \mathbf{x}_t \frac{p(\mathbf{x}_t|\mathbf{Z}_t)}{q(\mathbf{x}_t|\mathbf{Z}_t)} q(\mathbf{x}_t|\mathbf{Z}_t) d\mathbf{x}_t. \tag{12}$$

Its Monte-Carlo approximation is given as

$$\hat{E}(\mathbf{x}_t) = \sum_{n=1}^{N} \mathbf{x}_t^{(n)} \frac{p(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}{q(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(n)}). \tag{13}$$

In Equation (13), the probability for $\mathbf{x}_t$ is approximated as

$$\hat{p}(\mathbf{x}_t) = \sum_{n=1}^{N} \frac{p(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}{q(\mathbf{x}_t^{(n)})|\mathbf{Z}_t} \delta(\mathbf{x}_t - \mathbf{x}_t^{(n)}). \tag{14}$$

Since $\pi(\mathbf{x}) \propto p(\mathbf{x})$, $\pi(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)$ replaces $p(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)$. Then, $\frac{\pi(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}{q(\mathbf{x}_t^{(n)}|\mathbf{Z}_t)}$ is $w_t^{(n)}$ as in Equation (10), so that Equation (14) becomes an approximation of the true probability as

$$\hat{p}(\mathbf{x}_t) = \sum_{n=1}^{N} w_t^{(n)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(n)}). \tag{15}$$

To estimate the posterior probability in Bayesian tracking, importance sampling should be performed whenever measurements are available. Fortunately, the importance density factors into two parts dependent on the current time t and the previous time $t-1$ as

$$q(\mathbf{x}_t|\mathbf{Z}_t) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{Z}_t) q(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1}). \tag{16}$$

Then, the weight, $w_t^{(n)}$, is recursively described using the previous weight, $w_{t-1}^{(n)}$ at time $t-1$ and the current measurements as

$$\begin{aligned} w_t^{(n)} &\propto \frac{\pi(\mathbf{z}_t|\mathbf{x}_t^{(n)})\pi(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)})\pi(\mathbf{x}_{t-1}^{(n)}|\mathbf{Z}_{t-1})}{q(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}, \mathbf{Z}_{t-1}) q(\mathbf{x}_{t-1}^{(n)}|\mathbf{Z}_{t-1})} \\ &= w_{t-1}^{(n)} \frac{\pi(\mathbf{z}_t|\mathbf{x}_t^{(n)})\pi(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)})}{q(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}, \mathbf{Z}_{t-1})}. \end{aligned} \tag{17}$$

This sequential importance sampling is a key attribute of the particle filter. However, the particle filter still has other critical issues, such as degeneracy, the curse of dimensionality, the choice of a proposal function, etc., which will be briefly explained.

2.4.2 Degeneracy

In sequential importance sampling, degeneracy indicates a phenomenon whereby very few particles have high weights, but most of the particles have negligible weights after a few iterations. Since degeneracy wastes most particles, it becomes difficult to estimate a true posterior probability using the remaining few particles. There is a criterion to measure the degree of the degeneracy, called an effective sample size, N_{eff}, is

$$\hat{N}_{eff} = \frac{1}{\sum_{n=1}^{N} (w^{(n)})^2}. \tag{18}$$

For example, if all particles have the same weight, $\frac{1}{N}$, $\hat{N}_{eff} = N$. If only one particle has weight 1 but the other particles have 0, $\hat{N}_{eff} = 1$. Therefore, a small $\hat{N}_{eff}$ means severe degeneracy. The degeneracy should be predicted beforehand and avoided by using either a good proposal function or a resampling process.

2.4.3 Proposal Function

Since a proposal function determines how to distribute particles, a well-designed proposal function greatly improves the effectiveness of particles to describe a probability. The proposal function used the most frequently is a state update model, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ [21]. However, when a true measurement falls far from the estimated state, particles generated using the state update model will not be distributed well. Therefore, an optimum proposal function has the form of a posterior distribution shown as [17]

$$q(\mathbf{x}|\mathbf{x}_{t-1}^{(n)}, \mathbf{Z}_t)_{opt} \propto p(\mathbf{z}_t|\mathbf{x}_t^{(n)}) p(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}). \tag{19}$$

2.4.4 Resampling

Another solution for degeneracy is to resample particles whenever significant degeneracy occurs. This is straightforward although it sometimes causes another problem, sample impoverishment, which means that samples are not distributed diversely because many samples are repeated by resampling. However, resampling is very effective for reducing degeneracy. The basic idea of resampling is to eliminate samples that have negligible weights, and copy samples that have high weights, maximizing the effectiveness of the samples. After resampling, all samples have the same weight of $\frac{1}{N}$.

The resampling algorithm is implemented by calculating a cumulative probability density in Table 2.1. Then, a generic particle filter is shown as in Table 2.2, which includes the resampling.

Table 2.1: A resampling algorithm.

For n=1:
Initialize a cumulative density function (CDF): $c_1 = 0$
For n=2...N:
Construct a CDF using $w_t^{(n)}$: $c_n = c_{n-1} + w_t^{(n)}$
Generate one sample from uniform distribution: $u_1 \sim U[0, N^{-1}]$
For j=1...N:
Move along the CDF of u_j: $u_j = u_1 + \frac{(j-1)}{N}$
While $u_j > c_n$, $n = n + 1$
Keep the index of a parent for sample j : $j(n) = n$
For j=1...N:
Assign $\mathbf{x}_t^j = \mathbf{x}_t^{j(n)}$, $w_t^j = \frac{1}{N}$

2.4.5 Multiple Target Tracking

The curse of dimensionality is another issue in a generic particle filter. The curse of dimensionality indicates a problem caused by an exponential increase of a volume or a large dimension by adding extra spaces. This curse of dimensionality occurs when a standard particle filter is applied to multiple-target tracking. When a particle filter

Table 2.2: A genetic particle filter.

For n=1...N

Generate $\mathbf{x}_t^{(n)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{Z}_t)$.

Calculate importance weights for $\mathbf{x}_t^{(n)}$:

$w_t^{(n)} \propto w_{t-1}^{(n)} \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(n)})p(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)})}{q(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)},\mathbf{Z}_{t-1})}$

Normalize the importance weights $w_t^{(n)}$.

Calculate $\hat{N}_{eff}$ using Equation (18):

$\hat{N}_{eff} = \frac{1}{\sum_{n=1}^{N}(w^{(n)})^2}$

If resampling is needed, resample the particles and assign $w_t^{(n)} = \frac{1}{N}$.

tracks multiple targets, the dimension of its state space is multiplied by the number of targets. Then, one subspace for one target in the state space has a high weight, but the other subspaces for other targets in the same state space may have low weights. Then, the weight for this state space becomes small due to the subspaces having low weights. Therefore, as the number of targets increases, the required number of particles exponentially increases to achieve a certain performance in a standard particle filter. Orton [53] addressed this problem well and proposed applying partitioned sampling for this problem.

Partitioned sampling [47] is a strategy that divides the state space into two or more partitions and sequentially generates particles for each partition followed by an appropriate resampling operation. Orton partitioned each target as an individual partition and then applied importance sampling for each target independently. This algorithm is called an independent partitioned particle filter (IPPF), which is summarized in Table 2.3. The IPPF works well with fewer particles than a generic particle filter, when both are applied to multiple-target tracking. However, this is only true when accurate data association between measurements and targets is provided [8,53].

Table 2.3: An algorithm for the independent partitioned particle filter.

Define $\mathbf{x}_t = [\mathbf{x}_{1,t} \dots \mathbf{x}_{K,t}]^T$
At time t=0:
  Initialize all particles with initial values.
For time t > 0:
For k=1 ... K, n=1 ... N,
  Sample $\mathbf{x}_{k,t}^{(n)} \sim q_k(\mathbf{x}_{k,t} | \mathbf{x}_{k,t-1}^{(n)}, \mathbf{Z}_t)$
  Calculate the partition weight function,
  $\alpha_k^{(n)} = \frac{p_k(\mathbf{x}_{k,t}^{(n)} | \mathbf{x}_{k,t-1}^{(n)}) p_k(\mathbf{z}_t | \mathbf{x}_{k,t}^{(n)})}{q_k(\mathbf{x}_{k,t}^{(n)} | \mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)}$
  Normalize $\alpha_k^{(n)}$ for each partition
  Resample $\mathbf{x}_k^{(n)}$ with $\alpha_k^{(n)}$ and reindex $\mathbf{x}_{k,t}^{(n)}$
For n=1... N:
  $\mathbf{x}_t^{(n)} = [\mathbf{x}_{1,t}^{(n)}, \dots, \mathbf{x}_{K,t}^{(n)}]^T$
  Calculate the importance weights:
  $w_t^{(n)} = w_{t-1}^{(n)} \frac{p(\mathbf{z}_t | \mathbf{x}_t^{(n)}) p(\mathbf{x}_t^{(n)} | \mathbf{x}_{t-1}^{(n)})}{q(\mathbf{x}_t^{(n)} | \mathbf{x}_{t-1}^{(n)}, \mathbf{Z}_t)}$
  Normalize the importance weights $w_t^{(n)}$
  Resample the particles and assign $w_t^{(n)} = \frac{1}{N}$

2.5 Data Association

Data association between measurements and targets is always required in both single and multiple-target tracking. In single-target tracking, clutter creates the need for data association. In multiple-target tracking, even in the absence of clutter, there is inevitable ambiguity of assigning a measurement to one of multiple targets. This ambiguity is depicted in Figure 2.4. In Figure 2.4, z_1, z_2, z_3 and z_4 denote the measurements detected from sensors. The dotted lines in the figure show hidden, true associations between the measurements and targets, but there is no clue or information transmitted from the feature detection how each measurement is related to the target. Therefore, data association has to deal with the association between the unlabeled measurements and the targets. It also deals with cases of missing data and falsely detected data.

Since this research uses a non-linear Bayesian approach for the main tracking framework, data association also should be integrated into this Bayesian framework.

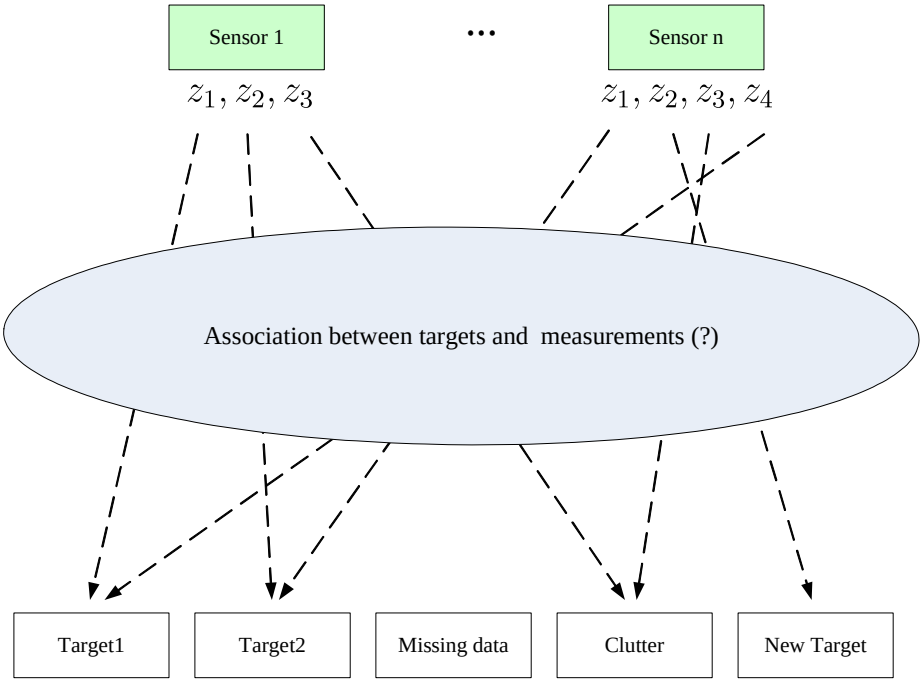


Figure 2.4: The ambiguity between targets and measurements.

The simplest method is a nearest-neighbor standard filter (NNSF) [6]. The NNSF predicts a measurement by using past measurements without looking at real measurements, which is called a predicted measurement. Around the predicted measurement in the measurement space, the NNSF draws a gate or validation region. This validation region is an area where a new measurement can be found with a high probability. The NNSF considers measurements inside the validation region as valid measurements. Then, as a true measurement, it chooses the nearest real measurement from the predicted measurement. This is a fairly reasonable approach, but there are cases when no measurement or multiple measurements exist in the region. If there are multiple measurements in the validation region, choosing the nearest measurement as a true measurement may discard other feasible measurements. Instead of choosing only one measurement for one target, a probabilistic data association filter, which keeps all feasible measurements to each target but reflects each feasible measurement's contribution as a probability, was developed.

The most representative probabilistic data association methods are a probability data association filter (PDAF)for a single target and a joint probability data association filter (JPDAF) and a multiple hypothesis filter (MHF) for multiple targets [6,68]. These methods use a gating of a NNSF as the first filtering step in order to reduce the number of valid measurements. Then, the JPDAF associates measurements to targets. It cannot deal with new targets, but it does generate a more feasible number of data association hypotheses. A MHF attempts to use all possible association hypotheses, including new targets, by associating a measurement to a target, but it suffers from the computational complexity caused by a much higher number of possible association hypotheses than JPDAF.

In order to apply these probabilistic data association methods to the framework of a particle filter, JPDAF was implemented based on discrete samples [45,56], and then Vermaark summarized this approach and called it MC-JPDAF in [68]. He also tried to drive data association methods directly from the particle filter framework [68].

This research will develop two data association methods in the particle filtering framework using multiple cameras and microphones, which are based on Vermaak's work [68]. One is MC-JPDAF, which is Monte-Carlo approximation of JPDAF. The other is an extension of IPPF with data association, called DA-IPPF here. This research will further improve the DA-IPPF and compare the performance of MC-JPDAF and DA-IPPF, when applied to acoustic speaker tracking, visual target tracking, and joint audio-visual tracking in subsequent chapters.

2.6 Sensor Fusion

Sensor fusion is the process of combining information from various sensors. Sensor fusion seeks synergistic effects using different, multiple sensors. One potential gain is to yield more robust and consistent results using redundancy and complementarity from different sensors [67]. In communication or data compression, redundant

information should be removed. In tracking, however, redundant information from multiple sensors ensures more robust tracking because all measurements inherently have some degree of uncertainty. Complementarity between multiple sensors also improves the robustness of tracking. For example, visual data helps to track a speaker in a room with high reverberation. Reverberation affects acoustic data greatly but has no effect on visual data. Therefore, using different kinds of sensors usually increases the robustness of tracking.

Another potential gain of sensor fusion is to yield new information [2, 22, 54]. For example, consider using stereo cameras. Since one camera image gives only $2D$ images of the environment, a point on the 2D image maps to a line in 3D. However, images from stereo cameras can yield a point in 3D.

The way humans learn about their surroundings is the most representative sensor fusion of all. Sensor fusion in humans is too natural to explain. However, it is not a simple task for a machine. A complicated mechanism has to be set up to make inferences using different information from different sensors. The most popular techniques are fuzzy logic, Dempster-Shafer reasoning, a neural network, and Bayesian approaches [28]. Hosein Nezhad [28] compared the performance of fuzzy, Dempster, and a Bayesian approach and concluded that the Bayesian approach was more appropriate for the authors' application of navigation and path planning in an occupancy grid map.

A Bayesian approach for sensor fusion is widely used because of its simple fusion mechanism. The fusion mechanism is statistical data fusion based on Bayes' theorem and a hidden Markov model. It is implemented by assuming conditional independence between measurements from multiple sensors when the hidden target states are given. If the measurement from sensor 1 is denoted by $\mathbf{z}_1$ and the measurement from sensor 2 is denoted by $\mathbf{z}_2$, the posterior probability for $\mathbf{x}$ is described by using a prior, $p(\mathbf{x})$, and a joint data likelihood, $p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x})$. Then, the conditional independence makes

the joint data likelihood the product of the two data likelihoods for each measurement as

$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2) = \frac{p(\mathbf{x})p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x})}{p(\mathbf{z}_1, \mathbf{z}_2)} \tag{20}$$

$$\propto p(\mathbf{x})p(\mathbf{z}_1|\mathbf{x})p(\mathbf{z}_2|\mathbf{x}). \tag{21}$$

This is the main mechanism of sensor fusion. Even when there are many similar or different sensors, the same rule applied in Equation (21) is applied.

According to how the measurements from the sensors are organized, the sensor fusion is categorized in two ways: centralized fusion or decentralized fusion [34, 42]. Centralized fusion is shown in Figure 2.5(a) and decentralized fusion in Figure 2.5(b). Centralized fusion simultaneously collects all measurements from sensors and uses them at one time, so its posterior probability can be evaluated as a product of the posterior probability from each sensor.

$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_n) \propto p(\mathbf{x})p(\mathbf{z}_1|\mathbf{x})p(\mathbf{z}_2|\mathbf{x})p(\mathbf{z}_3|\mathbf{x}) \ldots p(\mathbf{z}_n|\mathbf{x}). \tag{22}$$

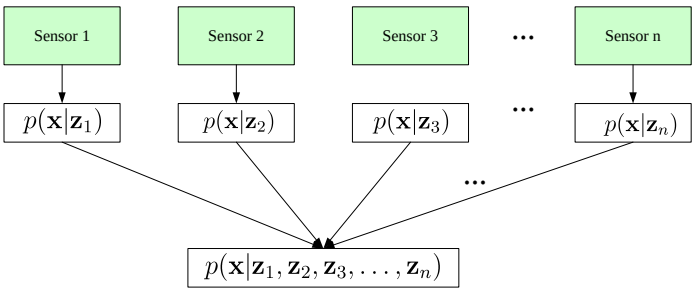
Centralized fusion can be used for a small video conference system where the small numbers of cameras and microphones are connected to one system and capture data synchronously, as in our research.

Decentralized sensor fusion collects and fuses the measurements from each sensor one by one in a certain order, so the posterior probability is described as

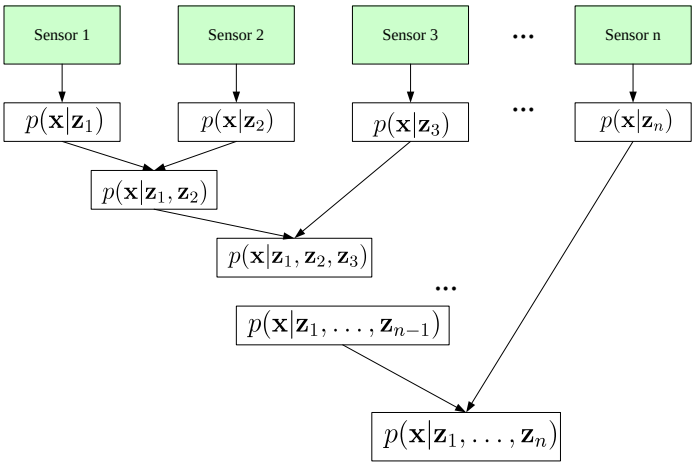
$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_n) \propto p(\mathbf{x}|\mathbf{z}_n)p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_{n-1}). \tag{23}$$

Decentralized fusion can be applied for a wireless network with thousands of nodes with different delays. In the decentralized method, each sensor calculates the posterior probability by using its own measurements and the posterior probability transmitted from the preceding sensors.

Considering sensor fusion for multiple targets using cameras and microphones, there are cases when only one person talks or when no one talks. In these cases, the



(a) Centralized sensor fusion



(b) Decentralized sensor fusion

Figure 2.5: The structure of the centralized and decentralized sensor fusion.

hidden state for the acoustic measurements is only a part of the state, and the sensor fusion in Equation (22) or Equation (23) cannot be applied directly. This problem can be solved using partitioned sampling, or it can be solved by data association without any additional processing. During data association, each measurement is tested and determined which substate the measurement belongs to. Therefore, an acoustic measurement is associated with its hidden state through data association. Then, sensor fusion is done using the data association.

CHAPTER III

DETECTION OF ACOUSTIC AND VISUAL FEATURES

3.1 Overview

Since the performance of target tracking greatly depends on the accuracy of the detected targets features, robust feature detection is necessary for good tracking. Feature detection usually consists of two parts: choosing appropriate features and developing algorithms to detect those features. This section describes acoustic-feature detection first followed by visual-feature detection. Each feature detection method has different requirements and challenges.

3.2 Detection of Acoustic Features

There are several requirements for acoustic-feature detection in this research. First and most important is that it should detect multiple sources. Second is that it must be used in a closed room environment, not in an open space, which means that the input acoustic signals are affected by reverberation as well as random noise. The third requirement is that it should be an appropriate acoustic feature for particle filtering. The last requirement is that it should use a small number of microphones, no more than six or seven.

With these requirements, both a direction of arrival (DOA) and a time delay of arrival (TDOA) were considered for acoustic features. The DOA is usually detected using beamforming, which was initially determined to be more appropriate for this research because it can detect multiple targets. However, since only one DOA is generated from one microphone array using a beamformer, DOA estimation needs at least two microphone arrays to track 3D locations in Cartesian coordinates. This

contradicts our requirement of using a small number of microphones. It also requires that the microphones be located within a spacing of $\frac{\lambda}{2}$ to prevent aliasing in the spatial domain [26]. Here, λ represents the maximum bandwidth of the input signal.

TDE methods using TDOA features are known to be limited to detect only a single source. However, they can localize multiple sources when TDOAs are used for measurements with data association as will be discussed in Chapter 4. In addition, since one TDOA is calculated from only a pair of microphones, not a whole array, using TDOAs requires far fewer microphones than beamformers. Therefore, this research determined that TDOAs were more appropriate features than DOAs.

3.2.1 TDOA Estimation

To define a model for a TDOA, the positions of all microphones are assumed to be known in advance. Then, the distance and the propagation time between a source and microphone i are described as

$$D_i \triangleq \sqrt{(x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2} \tag{24}$$

$$\tau_i \triangleq \frac{D_i}{c} \tag{25}$$

where (x_i, y_i, z_i) denotes the known position of microphone i, (x_s, y_s, z_s) denotes the unknown 3D position of a source, and c is the speed of sound in air.

Then, a TDOA between a source and microphone i and the source and microphone j is defined using the distances between the source and each microphone. d_{ij} is the distance difference between D_i and D_j, and τ_{ij} is the TDOA.

$$d_{ij} \triangleq D_i - D_j \tag{26}$$

$$\tau_{ij} \triangleq \frac{D_i - D_j}{c} = \tau_i - \tau_j. \tag{27}$$

In reality, the TDOA is quantized to an integer because it is estimated as the delayed number of samples between two digital audio signals, captured at a sampling frequency F_s, as in Equation (28). $\lfloor$ in Equation (28) indicates that the value in the

parentheses is an integer value.

$$\hat{\tau}_{ij} = \lfloor (\tau_{ij} F_s)[samples]. \tag{28}$$

To estimate the TDOA from real speech signals, the phase transform (PHAT) [39] is applied. The PHAT is similar to a generalized cross-correlation (GCC) method [39] but uses only phase information from the input signals because the phase information carries all the delay information.

$$\tau \triangleq arg \max_{\tau} \int_{-\infty}^{+\infty} \frac{X_i(f) X_j^*(f)}{|X_i(f) X_j^*(f)|} e^{j\pi f \tau} df \tag{29}$$

where $x_i(t)$ and $x_j(t)$ are the signals at microphones i and j in the time domain, and $X_i(f)$ and $X_j(f)$ are the corresponding signals in the frequency domain.

In an ideal free field environment, only one single, direct path exists for the propagated signals, as in Figure 3.1(a). In this ideal free field, the signal captured in microphone i is an attenuated, delayed version of the direct source signal. This signal in the time domain and the corresponding signal in the frequency domain are described as

$$x_i(t) = \alpha_i s(t - \tau_i) \tag{30}$$

$$X_i(f) = \alpha_i e^{-2\pi \tau_i f} S(f) \tag{31}$$

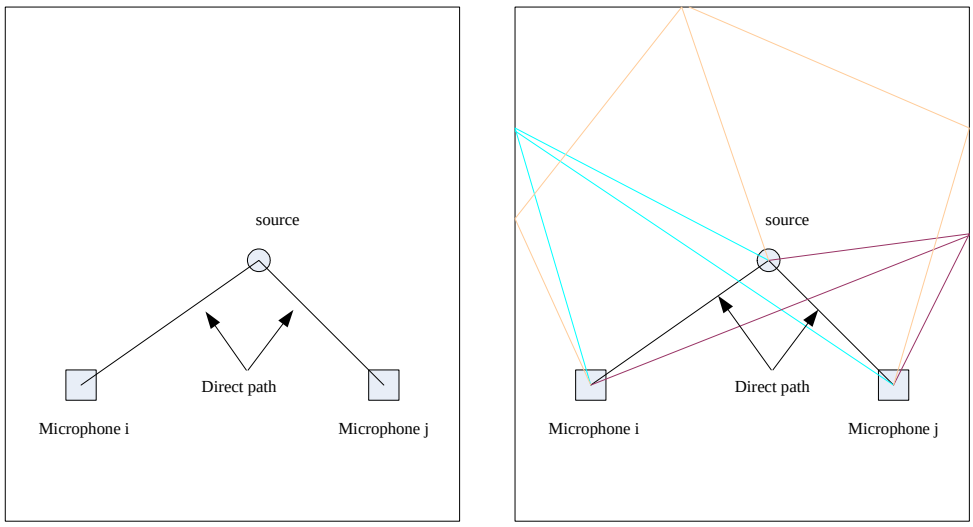
where $s(t)$ is a source signal, and α_i is an attenuation parameter.

By using the equations above, Equation (29) can be simplified to

$$\begin{aligned} \tau_{ij} &= arg \max_{\tau_i - \tau_j} \int_{-\infty}^{+\infty} \frac{\alpha_i e^{-2\pi \tau_i f} S(f) \alpha_j e^{2\pi \tau_j f} S(f)^*}{|\alpha_i \alpha_j S(f) S(f)^*|} e^{j\pi \tau f} df \\ &= arg \max_{\tau_i - \tau_j} \int_{-\infty}^{+\infty} e^{-j2\pi f(\tau_i - \tau_j)} e^{j2\pi \tau f} df. \end{aligned} \tag{32}$$

The PHAT performs worse than GCC for periodic, narrow-band signals, but outperforms GCC in speech.

In a reverberant environment, as in Figure 3.1(b), the signal propagates through one direct path and numerous indirect paths. Therefore, Equation (31), the signal



(a) In an idea free field environment (b) In reverberant environment

Figure 3.1: Signal transmission in an ideal free field environment and in a reverberant environment. The rectangle indicates a closed room.

model for propagation based on an ideal free field model, is no longer valid. To detect TDOAs that are tolerant to reverberation, multiple TDOAs are detected through the use of the PHAT method. The detected multiple TDOAs are assumed to include both TDOAs from direct paths of the real sources and TDOAs caused by reverberation. While the TDOAs caused by reverberation do not have temporal consistency, the TDOAs from true sources do.

Random noise in a room is also considered because the random noise degrades general performance for the cross-correlation of two signals. Therefore, it affects the accuracy of detected TDOAs significantly.

Incorrect TDOAs and missing TDOAs caused by reverberation and random noise can be solved by data association, as discussed in Chapter 4, which figures out which TDOA is valid by exploiting temporal consistency.

3.3 Simulation for Acoustic Feature Detection

The simulation in the section is designed for four purposes. One is to show the accuracy of TDOA estimation with respect to the positions of the microphones. The

second is to show that TDOA can detect multiple sources. The third is to evaluate the performance of TDOA detection according to different signal-to-noise ratios. The last is to evaluate the performance of TDOA detection according to different room reverberation times.

3.3.1 Simulation Environment

Figure 3.2 shows the floor plan of a conference room. The size of the room is approximately 590[cm] x 360[cm] x 270[cm]. The reference point of the room, $(0, 0, 0)$, is the left and bottom corner of the room as indicated in the figure. The microphones are assumed to be located in one of two distributions: a centralized array and a distributed array. In Figure 3.2, the numbers in circles in the center of the room show the locations of all the microphones in the centralized array. The numbers along the walls show the locations of all the microphones in the distributed array. Table 3.4 shows the correct positions of the microphones. The numbers in parentheses in Table 3.4 are values on the x, y, and z axes.

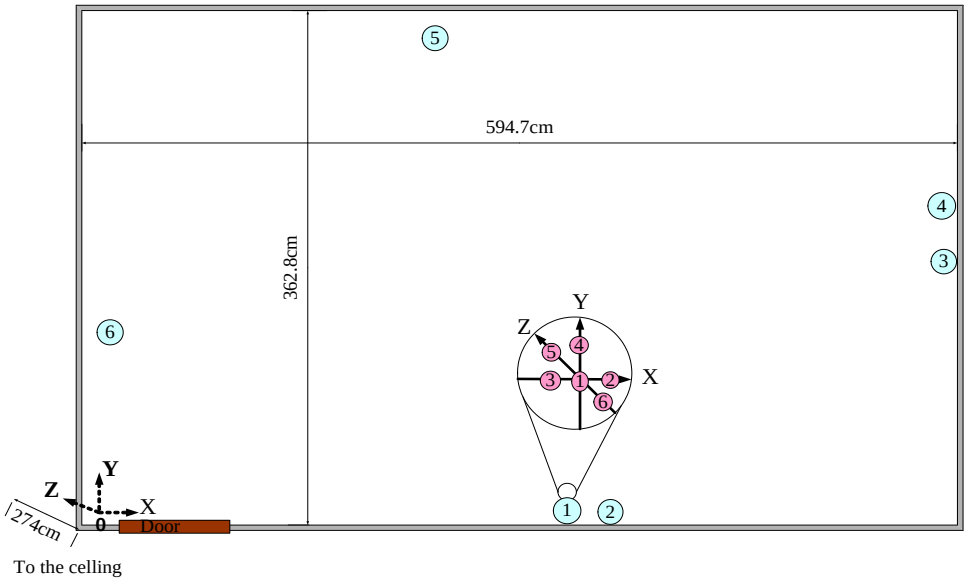


Figure 3.2: The locations of multiple microphones in a room.

For simulation, two different voices of a male and a female are captured at a

Table 3.4: The locations of microphones in an array.

microphone #	Centralized array	Distributed array
1	(300, 30,120)	(300, 0,100)
2	(330, 30,120)	(330, 0,270)
3	(270, 30,120)	(590,200, 30)
4	(300, 0,120)	(590,230,270)
5	(300, 30,150)	(200,360, 30)
6	(300, 30, 90)	(0,200,250)

sampling rate of 16 KHz and used as sound sources. These signals are shown in Figure 3.3. The speech signals are delayed and attenuated according to the distances between the locations of the microphones and the source(s) for simulations using the ideal free field model in Equation (31):

$$x_i(t) = \frac{D_1(s(t))}{D_i(s(t))} s(t - \tau_i(s(t))),$$

where $x_i(t)$ is the signal at microphone i, and $D_i(s(t))$ is the distance between microphone i and source $s(t)$ in Equation (25). To implement attenuation in this simulation, D_1 is used as a reference distance. If the distance between microphone i and the source is longer than the reference distance (i.e., the distance between microphone 1 and the source), the amplitude of $x_i(t)$ is attenuated. If the distance between microphone 1 and the source is shorter than the reference distance, the amplitude of $x_i(t)$ is magnified. For the simulation of two speakers, both male and female speech signals are delayed, attenuated, and then added as

$$x_i(t) = \frac{D_1(s_1(t))}{D_i(s_1(t))} s_1(t - \tau_i(s_1(t))) + \frac{D_1(s_2(t))}{D_i(s_2(t))} s_2(t - \tau_i(s_2(t))).$$

For the simulation of reverberation, we use Matlab program released by Lehmann [43]. His program can generate room impulse responses depending on the size of a room, the positions of multiple microphones, different reverberation time. The research generates room impulse responses of $T_{60} = 0.2$[sec] and $T_{60} = 0.5$[sec] using the distributed array. Then, these room impulse responses are convolved with the two speech signals.

Figure 3.3: Two sample speech signals for simulation.

3.3.2 Accuracy of TDOA according to Microphone locations

Unlike beamformers, TDOA estimation does not require that all microphones be located within a spacing of the half length of the maximum bandwidth of an input signal (i.e. $d \leq \frac{\lambda}{2}$). This is because TDOA estimation is performed in the time domain, not in the spatio-temporal domain. Therefore, TDOA estimation permits more freedom to place microphones. To maximize the performance of TDOA estimation in a room, this simulation tries two different microphone configurations: a centralized array and a distributed array in Table 3.4.

A centralized array is known to achieve good performance in the near field around the array but worse performance in the far field [31]. This indicates that if this research uses a centralized array, the tracking performance can be inaccurate in the far field. This is why the research tries both a centralized array and distributed array.

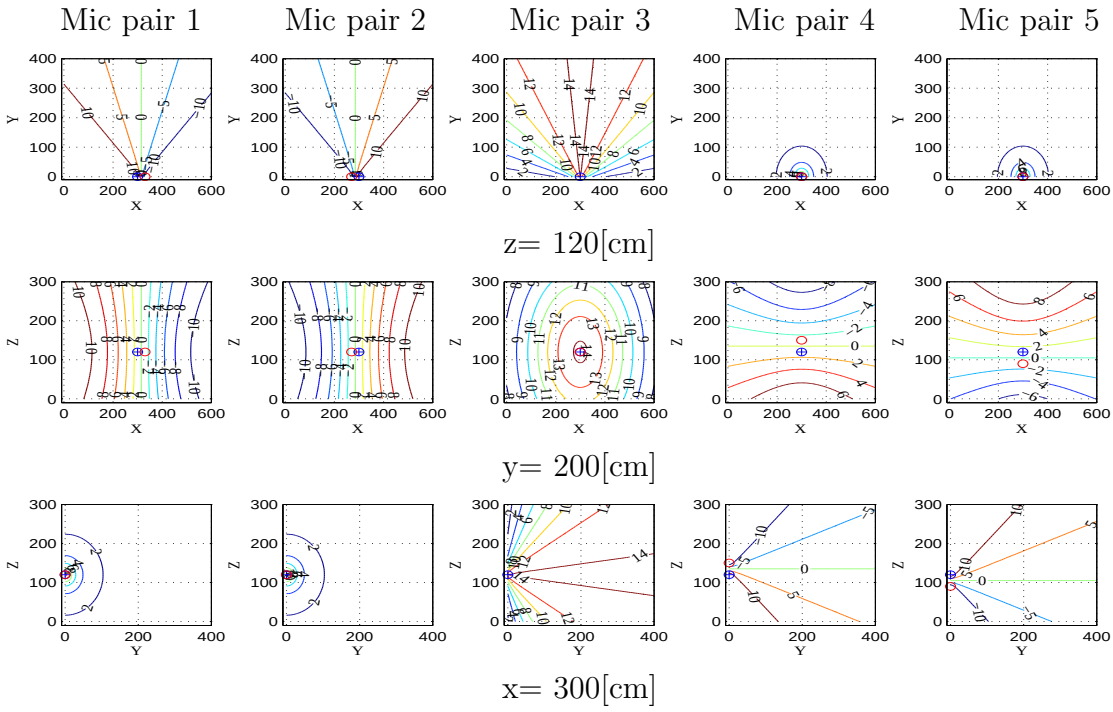
The reason the centralized array has inaccurate performance in a far field can be explained with TDOA contour maps for the room. Figure 3.4 shows the TDOA contour maps in different 2D planes (x-y, x-z, and y-z planes). The other remaining variable for these planes is fixed with a specific value given in the figure. For example,

the first row of Figure 3.4(a) shows contour maps in the x-y plane when z is fixed at 120[cm]. From left to right, each sub-figure shows a TDOA contour map for microphones 1 and 2, 1 and 3, 1 and 4, 1 and 5, and 1 and 6, which are labeled Mic pair 1, Mic pair 2, Mic pair 3, Mic pair 4, and Mic pair 5. Using six microphones, the maximum possible number of microphone pairs is $C(N^o, 2) = C(6, 2) = \frac{6!}{4!2!} = 15$, but this number of microphone pairs greatly increases the computational complexity at initialization and data association. Therefore, only five microphone pairs are used here.

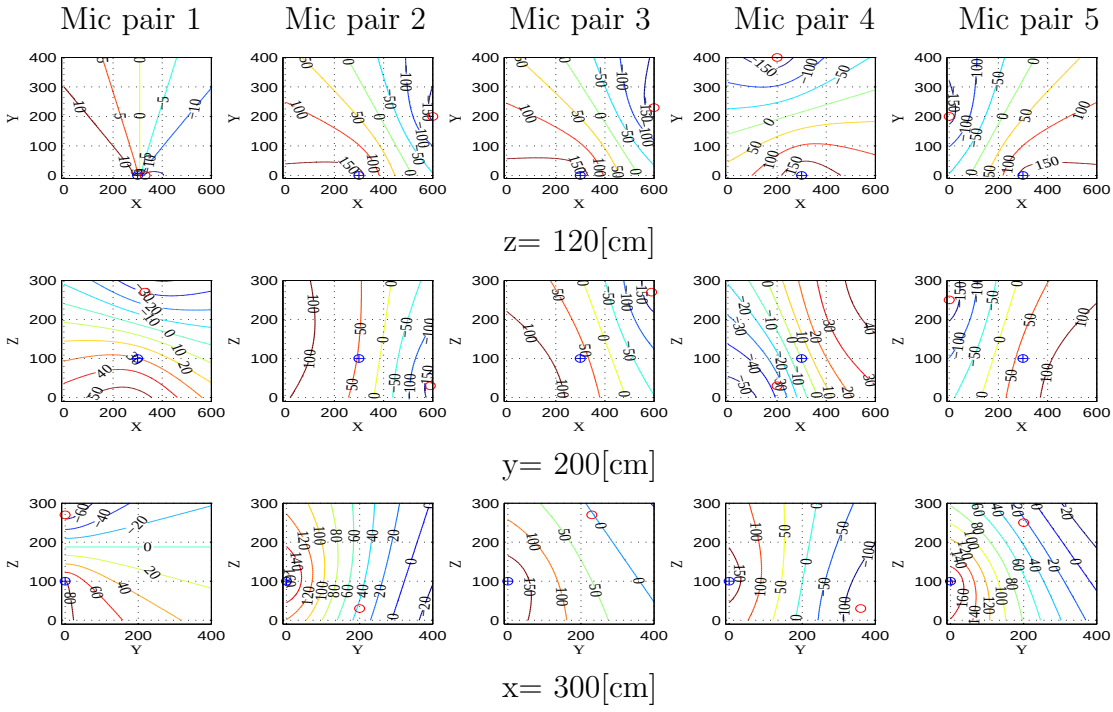
The localization of a source using multiple TDOA measurements is carried out to find the best intersection of all TDOAs corresponding to the source. Using the TDOA contour maps of the first row of Figure 3.4(a), the intersection of the TDOAs of a source in the far field of the room is represented as shown in Figure 3.5. As shown in the contour maps of Mic pair 4 and Mic pair 5, the TDOAs are the same in the back half of the room. This indicates that the TDOAs from these two pairs cannot contribute to the localization at all, but that the TDOAs from the first three pairs do. However, the overlapping area of three TDOAs from the first three sensor pairs is less like a point and more like a long ellipse. This ellipse gives a good idea only of the direction, not of the 3D position of the source. When the same scenario is applied to the distributed array, the intersection of five TDOAs using Figure 3.4(b) is more like a point than an ellipse. This is because, in the distributed array, the TDOAs are quite dense and distinct all around the room unlike the centralized array. Therefore, using the distributed array provides more consistent, better localization than using the centralized array.

3.3.3 Detection of Multiple TDOAs

The purpose for the simulation in this section is to check the accuracy of detected TDOAs according to different signal-to-noise ratios. It is assumed that there are two



(a) The centralized array



(b) The distributed array

Figure 3.4: TDOA contour maps in different 2D planes.

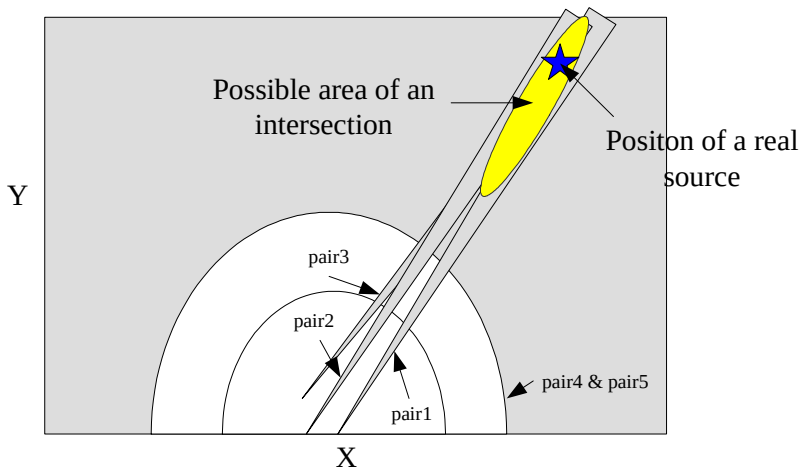


Figure 3.5: The overlapping area of three TDOAs from the first three sensor pairs for a single source in the far field in the x-y plane using the centralized array.

speakers moving as in Figure 3.6. The blue line shows Target 1, and the red line shows target 2. "St" shows the starting position, and "Ed" shows the ending position of each target. Target 1 moves from $(270, 260, 230)$ to $(400, 150, 120)$, and target 2 moves from $(350, 100, 170)$ to $(200, 150, 100)$ at a constant speed. The trajectories of these two targets will be used for the simulation of visual tracking and audio-visual tracking for performance comparison in the following chapters.

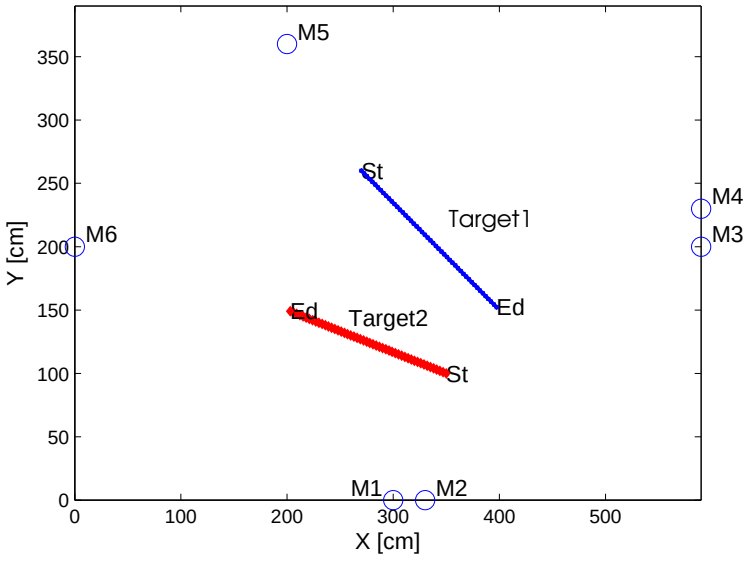
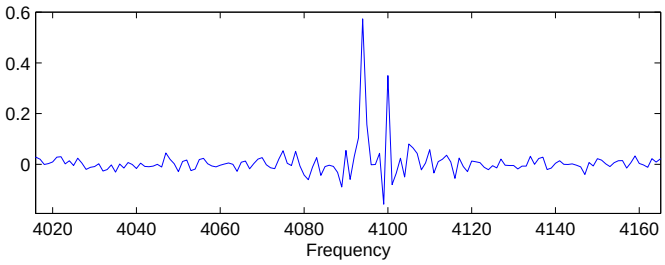


Figure 3.6: Trajectory of two speakers in the x-y plane for the simulation.

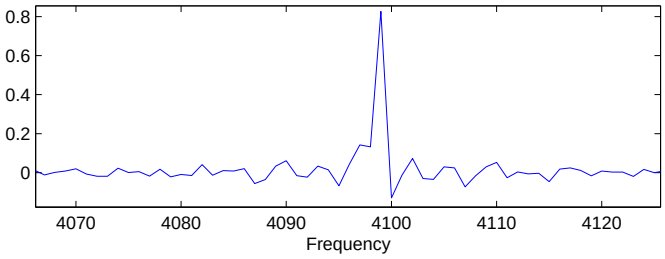
To estimate TDOAs using PHAT in the reverberant environment with noise, multiple peaks from the cross-correlation in Equation (32) are selected. However, choosing an optimum number of peaks can be tricky. When the number of targets is known, it is reasonable to select the same number of peaks as targets, but it may result in a loss of real peaks. This is because the peaks resulting from the indirect paths of one source caused by reverberation might dominate the peaks generated by the direct path of another source.

It may seem that it would be possible simply to set a certain threshold to detect the number of salient peaks. However, this approach is also unreliable because some peaks from true sources do not appear as distinct, as shown in Figure 3.7, which shows two examples of the cross-correlation in a different time frame with two sources. Figure 3.7(b) shows only one peak even though there are two sources, while Figure 3.7(a) shows two distinct peaks. To detect true TDOAs such as those in Figure 3.7(b) using thresholding, the threshold should be lower, but setting it lower can result in more false alarms. After numerous attempts to select the peaks using a fixed number or using thresholding, it was observed that using a fixed number obtains better results than thresholding. In a Gaussian noise environment or with a short reverberation time like $T_{60} = 0.2$, the missing data are minimized when the number of peaks chosen is the same as the number of targets, but with a reverberation time of $T_{60} = 0.5$, the missing data are minimized when the number of peaks chosen is one or two more than the number of targets.

Figure 3.8 shows the detected TDOAs using the centralized array and the distributed array in the scenario of Figure 3.6. Here, two peaks are chosen from each audio frame. The blue lines represent the true values of the TDOAs, and the red dots indicate detected TDOAs. Clearly, using the distributed array results in more correct detection and fewer false alarms and missing data than using the centralized array.

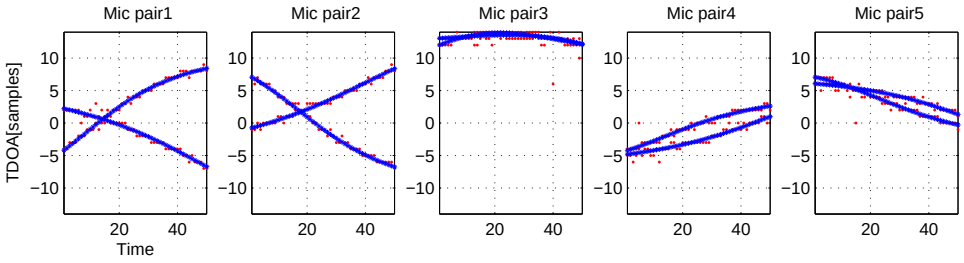


(a) Two distinct peaks for two speakers

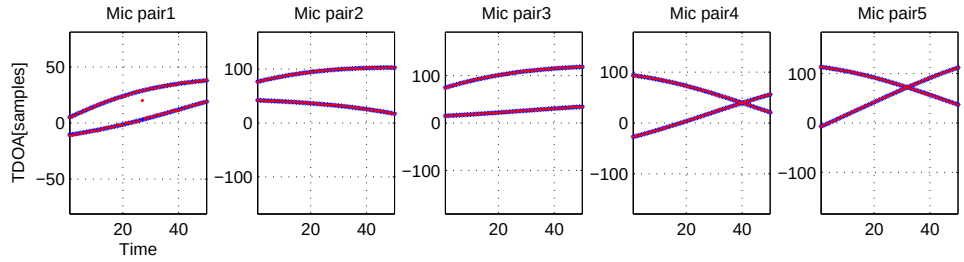


(b) One distinct peak for two speakers

Figure 3.7: Examples of cross-correlation.



(a) Using the centralized array



(b) Using the distributed array

Figure 3.8: The results of estimated TDOAs using different microphone arrays.

Another purpose of the simulation in this section is to evaluate performance degradation according to different signal-to-noise ratios. Here, the distributed array is used, and white Gaussian noise is added. Given the desired SNR, the variance of the noise

is calculated as

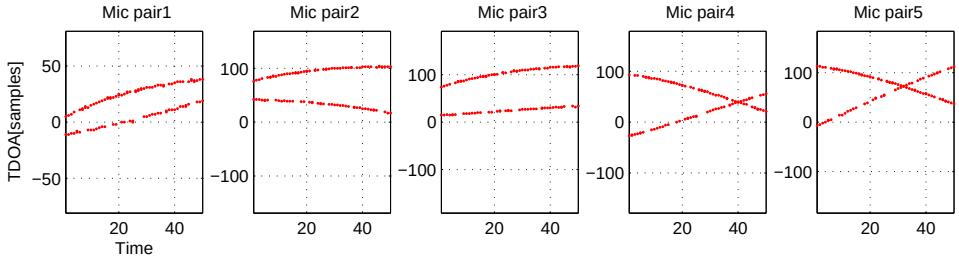
$$SNR = 10 \log \frac{\bar{s}^2(t)}{\bar{n}^2(t)}. \tag{33}$$

The estimated TDOAs according to different signal-to-noise ratios are demonstrated in Figure 3.9. As the SNR decreases from 30 dB to 10 dB, the missing data rates clearly increase. At 20 dB as in Figure 3.9 (b), the missing rate is around 0.13. In Figure 3.9 (c), even though more peaks are detected, the missing data rate does not improve much. In Figure 3.9 (d), at 10 dB, the missing data rate is around 0.3. When more peaks are detected, as in Figure 3.9 (e), more false alarms are generated, but the missing data rate does not decrease. The rate of false alarms in Figure 3.9 (e) is around 0.15. The environment of SNR = 10 dB is quite noisy. Generally, the noise in a closed environment is better than = 10 dB.

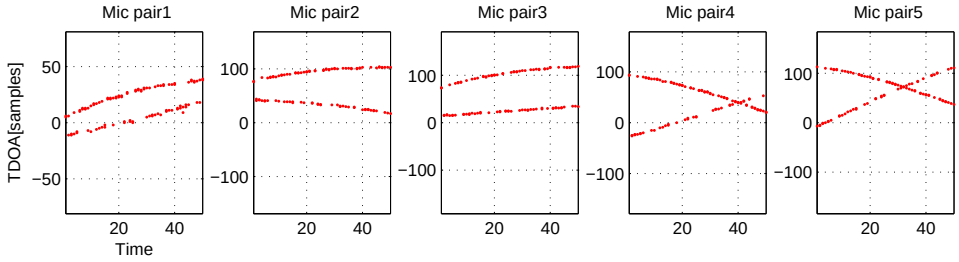
The estimated TDOAs according to different reverberation times are demonstrated in Figure 3.10. At reverberation time $T_{60} = 0.2$, the detected TDOA is quite accurate most of time, and the missing data rate is not significant. At reverberation time $T_{60} = 0.5$, the missing data rate is significant, more than 50%, when two peaks are chosen. When four peaks are chosen, the missing rate decreases slightly, especially at Mic pair 5.

3.4 Detection of Visual Features

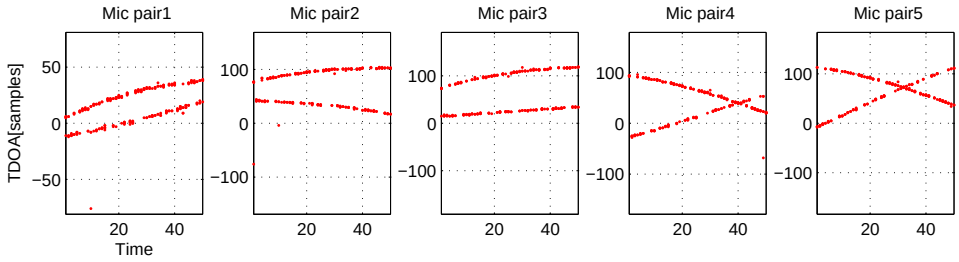
Visual-feature detection finds the appropriate visual features of multiple people using close-range cameras and develops algorithms to detect the visual features. Since the images from close-range cameras give the upper body parts of people, faces and heads can be the most appropriate visual features. However, when multiple people are moving in an irregular manner, while an acoustic source can be still modeled as a point source, visual features such as faces and heads should be modeled as non-rigid objects. The detection of these non-rigid faces/heads is not an easy problem. In addition, since this research uses multiple cameras, it requires considerable computation to process



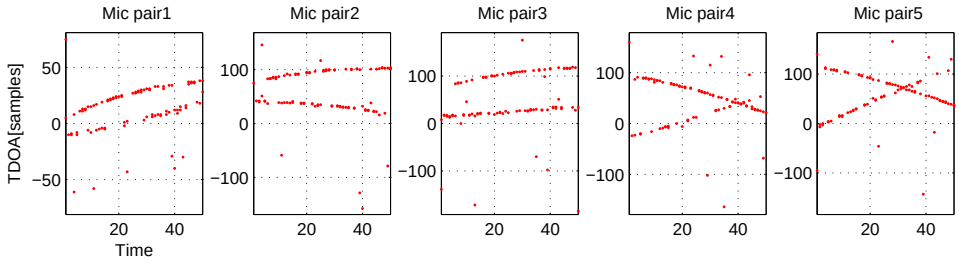
(a) Choosing two peaks, snr= 30 dB



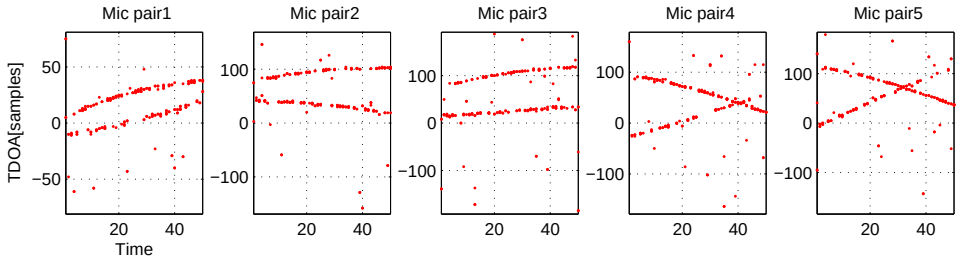
(b) Choosing two peaks, snr= 20 dB



(c) Choosing three peaks, snr= 20 dB

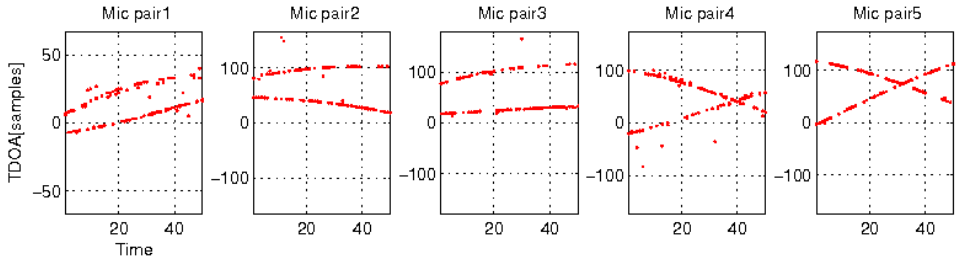


(d) Choosing two peaks, snr= 10 dB

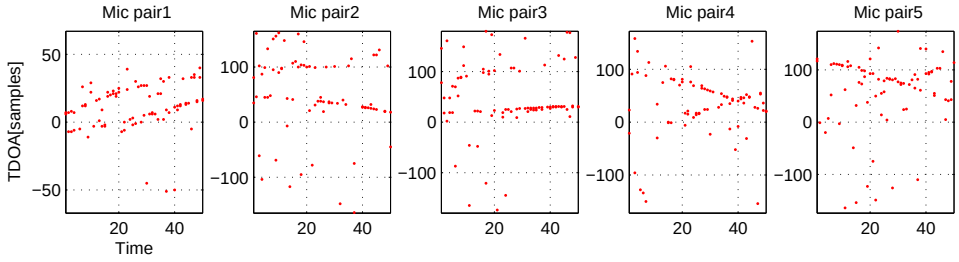


(e) Choosing three peaks, snr= 10 dB

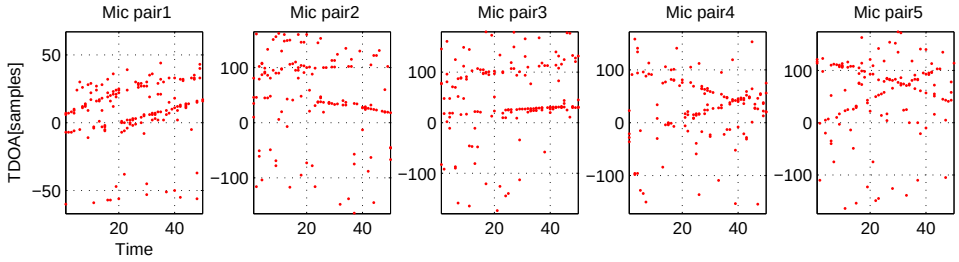
Figure 3.9: The estimated TDOAs according to different signal-to-noise ratio.



(a) Choosing two peaks, $T_{60} = 0.2$[sec]



(b) Choosing two peaks, $T_{60} = 0.5$[sec]



(c) Choosing four peaks, $T_{60} = 0.5$[sec]

Figure 3.10: The estimated TDOAs according to different reverberation times.

the multiple camera images. To reduce the computational complexity, this research uses small images such as 320 (H) x 240 (V). Therefore, visual-feature detection is also required to work with a low-resolution image.

First, skin-color detection was applied and evaluated for our test images. Because of its limitations described below, it was replaced by Viola-Jones detector as our face/head detection method. To overcome the low detection rate of this Viola-Jones detector for non-rigid faces/heads in time, video-based algorithms were tried. A mean-shift algorithm is applied to our video sequences, and then we propose using motion detection using corner detection/matching.

In addition, since multiple cameras have the same target(s), there is correlation between the targets from multiple cameras. This means that the detected features in one camera can give a hint of the location of the features to the other cameras. The simplest relation between a pair of cameras is epipolar geometry, which is expressed in a fundamental matrix. In this research, we simply use this point-to-line correspondence using epipolar geometry to determine the robustness of detected features.

3.4.1 Detection of Skin Color

In numerous studies in the literature, skin features have been used to detect faces [14, 27,30,54,55,67]. Past research has revealed that face color differences among different races is caused by a difference of luminance, not chrominance [74]. Therefore, the chrominance of skin color can be an invariant feature for the detection of faces. Since an algorithm for skin color detection is applied to each pixel, it is also an appropriate feature for the detection of non-rigid faces. However, because the area detected as skin can include bare arms and legs as well as faces, another algorithm should be applied to separate faces from the other skin color areas.

For skin color detection, a likelihood ratio test [27, 44] can be applied. This test calculates the data likelihood for skin and non-skin by using skin training data and

non-skin training data. Then, using the simple hypotheses that a pixel represents either skin or non-skin, the likelihood ratio test classifies each pixel as skin or non-skin. Figure 3.11 shows one result of skin color detection with the likelihood ratio test. The left sub-image in Figure 3.11 is an original image, and the middle image is a binary image. In this binary image, pixels detected as having skin color are white and pixels detected as having non-skin color are black. The right image shows the skin pixels in the original image. Most of skin parts, such as faces, necks, and hands, are correctly detected as skin, but the yellowish and pinkish wall is also detected as skin. The brown doors and tables are also detected as skin even though they are not shown here.

(a) Original image (b) Binary skin image (c) Detected skin-color image

Figure 3.11: One result of a skin color test using a likelihood ratio test.

Even though skin pixels in this test image are detected accurately, we cannot use skin color as a feature to detect faces, because skin feature detection produces high falsely detected data rate from background areas, such as walls and doors, that may have colors similar to those present in skin. Moreover, when a camera shows the heads of people behind, the skin color cannot detect any object. Therefore, luminance data can work better than chrominance in general, so we use a Viola-Jones detector as its starting point.

3.4.2 Initialization using Viola-Jones Detectors

The Viola-Jones detector [69] was initially developed to detect a frontal face. To detect multi-view faces, the Viola-Jones detector is trained with multi-view faces

captured from different angles [72, 75]. If the multi-view face training data are captured at each 10°, nine Viola-Jones detectors are needed, which requires considerable computation. Therefore, instead of training the Viola-Jones detectors with multi-view faces at different angles, this research uses three Viola-Jones detectors to detect three features : a frontal face, a face profile, and an upper body, which complementarily work with each other.

Figure 3.12(a) demonstrates the detection results of applying the three Viola-Jones detectors to relatively large faces. The red, blue, and yellow rectangles show the detected objects using the Viola-Jones detector for front face, face profile, and upper body features. The left image in Figure 3.12(a) detects one frontal face in a wrong place. The middle image in Figure 3.12(a) shows perfect detection by using all three detectors. The right image loses one face completely after the person rotates her face.

Figure 3.12(b) demonstrates the results when the sizes of faces in a test image are quite small, around 20 by 20 pixels. In this case, even though the faces are clearly frontal faces, they cannot be detected because the small faces lose details. The Viola-Jones detector for the upper body works in this case. Even though the Viola-Jones detectors for frontal faces and face profile do not detect the person, the Viola-Jones detector for upper body does detect the upper body of the person. The Viola-Jones detector for upper body also detects the upper body from behind. If only one upper body is detected, the location and size of a face/head can be approximated using the location and the size of the upper body. However, the three Viola-Jones detectors do not satisfactorily detect non-rigid faces/heads in time. In addition, since the Viola-Jones detector is frame-based feature detection, there is a possibility of reducing computational complexity when it is combined with a video-based approach.

Therefore, this research uses the three Viola-Jones detectors using frontal face, face profile, and upper body features at initialization. After the initial position of

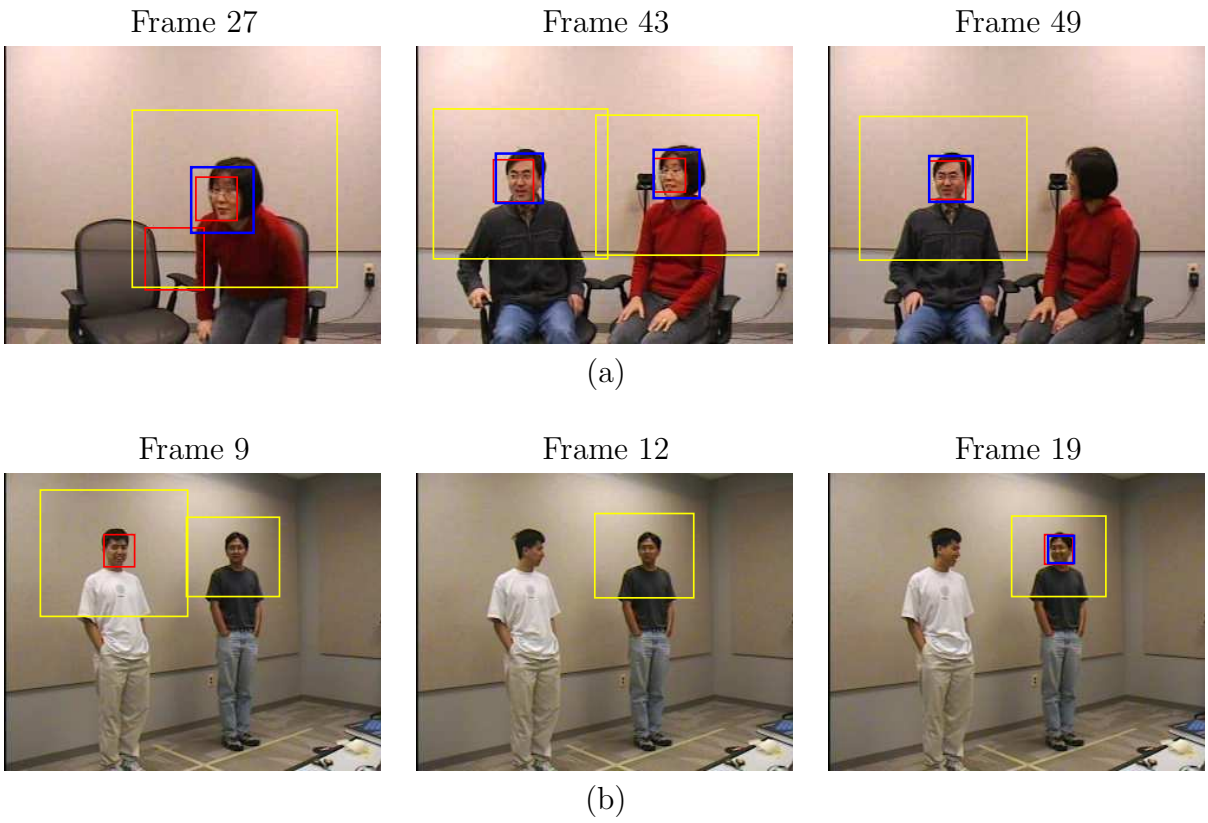


Figure 3.12: One result using three Viola-Jones detectors for a frontal face, a face profile, and upper body features.

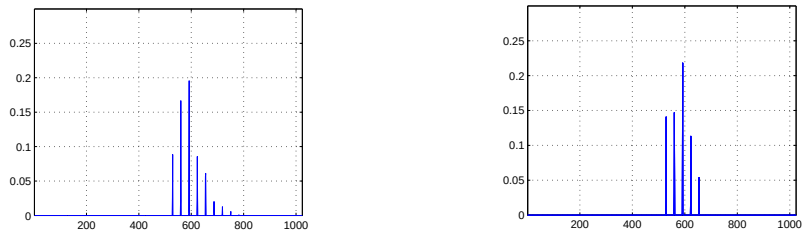
each object is found, the objects are detected using a video-based approach, which is explained below.

3.4.3 Mean Shift Algorithm

As a video-based approach, a mean-shift algorithm using color proved to be applicable for non-rigid object detection [15, 16, 59]. Even though the face detection using skin color was proved not to be applicable to our room environment, the mean-shift algorithm using color shows different results because it is performed based on a region, not on a pixel. The region including a face also includes hair and clothes, so the mean-shift algorithm works even with similar skin-color background clutter. However, the mean-shift algorithm becomes worse when the object moves quickly against a similar color background.

This research implements the mean-shift algorithm described in [16]. The results of applying the mean-shift algorithm to our test images are demonstrated in Figure 3.13 and Figure 3.14. The test images have two people. Figure 3.13 shows the initialization results using the three Viola-Jones detectors. The first person is detected at frame 22, and the second person is detected at frame 40. The kernel density is estimated from the pixels inside each rectangular window. Even though both objects are Asians with similar skin color and hair color, the kernel densities are quite different as in Figure 3.13(c) and (d). The kernel densities are calculated in a 32 x 32 window of the CbCr-plane and drawn after bearing changed into a column-based vector.

(a)Object 1 at frame 22 (b)Object 2 at frame 40



(c) The kernel density of object 1 (d) The kernel density of object 2

Figure 3.13: Initial positions of two targets detected using Viola-Jones detectors for a mean-shift algorithm and the kernel density of each object derived from the region of the red box.

Figure 3.14 demonstrates the final positions of the faces. The initialization occurs only once for each object, as in Figure 3.13. The sizes of the objects are fixed as the sizes of the initial faces, as in Figure 3.13. Figure 3.14 also shows the Bhattachayya coefficients for each target, $\rho_{k,n}$, for object k at frame n.

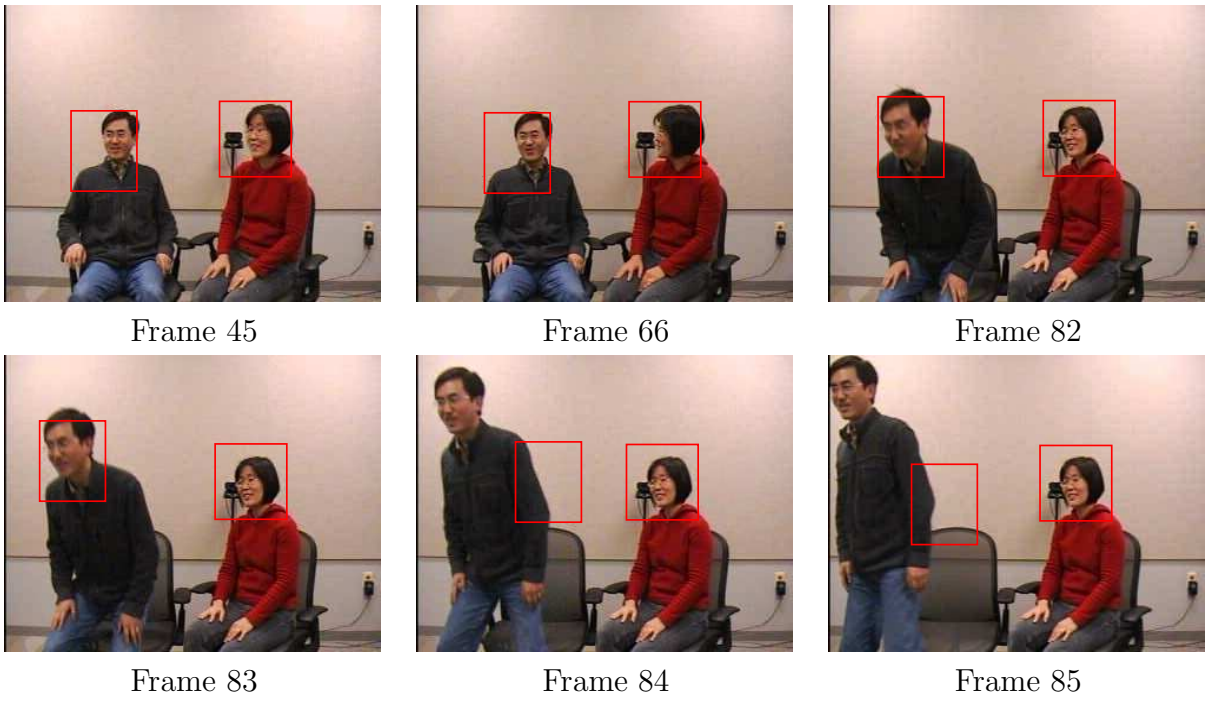


Figure 3.14: One result from the mean-shift algorithm applied to our test sequence($\rho_{1,45} = 0.987, \rho_{1,66} = 0.955, \rho_{1,82} = 0.964, \rho_{1,83} = 0.97, \rho_{1,84} = 0.970, \rho_{1,85} = 0.962, \rho_{2,45} = 0.984, \rho_{1,66} = 0.984, \rho_{1,82} = 0.994, \rho_{1,83} = 0.99, \rho_{1,84} = 0.823, \rho_{1,85} = 0.834$).

Even though the mean-shift algorithm using color shows a distinct kernel density for each object, the similar skin-color background hinders the accurate detection of faces. As shown in the first row in Figure 3.14, due to the similar skin-color background, the detected positions of object 2 are not exactly the center of the object even though they have high Bhattachayya coefficients. The worst result of the mean-shift algorithm applied to our test images is that, when object 2 moves quickly at frame 84 and 85, the mean-shift algorithm does not converge to the right position. Therefore, we tries a new feature, a corner, based on luminance data in video sequences, instead of improving the mean-shift algorithm because the mean-shift algorithm using color is inherently limited when the background color is skin-like.

3.4.4 Motion Estimation using Corner Detection/Matching

The mean-shift algorithm using color cannot detect a fast moving face over a similar skin-like background. The cause of the object being lost after it is detected is motion

in the object. This means that an object can be followed only if the motion of the object can be correctly detected. Consequently, the problem that needs to be solved becomes how to capture the global motion of the object.

As a method for motion estimation, a block-matching algorithm is the most representative algorithm and is used in video codecs such as MPEG-1, MPEG-2, and H.263, but this algorithm is based on blocks, not a single object. Methods for global motion estimation can detect the motion of the whole image, like that of a camera panning across an area instead of focusing on the motion of a specific object. Therefore, to detect the motion of a non-rigid object accurately, this research considers a totally different feature, a corner.

Corners are pixels that have strong luminance changes in orthogonal directions. The corners [24] are invariant to rotations, small scale changes, affine transformations, and small illumination changes [51]. They are used for random sample consensus (RANSAC), automatic homography, automatic image stitching [25], and non-rigid object tracking [49, 61]. This research proposes that the corners can be used as features for the motion detection of non-rigid objects. Motion detection using corners is composed of two parts: corner detection and motion estimation from the detected corners.

Among several corner detectors, a Harris corner detector is used most frequently [24, 60]. The Harris corner director was proposed to improve Moravec's corner detector [60] by using auto-correlation. Denoting an image intensity by $I(u, v)$ in the image domain and the intensity change in the image by a shift (x, y), E is given by

$$E(x, y) = \sum_{u,v} w(u, v)|I(u + x, v + y) - I(u, v)|^2 \tag{34}$$

where $w(u, v)$ is a smooth circular window like a Gaussian density. The shifted image $I(u+x, v+y)$ is approximated analytically using a Taylor expansion, and then

Equation (34) becomes

$$E(x,y) = \sum_{u,v} w(u,v)(xI_x(u,v) + yI_y(u,v))^2 \tag{35}$$

$$= \sum_{u,v} w(u,v)(x^2 I_x^2(u,v) + y^2 I_y^2(u,v) + 2xyI_x(u,v)I_y(u,v)) \tag{36}$$

$$= \begin{bmatrix} x & y \end{bmatrix} \begin{pmatrix} \sum_{u,v} w(u,v) I_x^2(u,v) & \sum_{u,v} w(u,v) I_x(u,v) I_y(u,v) \\ \sum_{u,v} w(u,v) I_x I_y(u,v) & \sum_{u,v} w(u,v) I_y^2(u,v) \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{37}$$

where $I_x(u,v)$ and $I_y(u,v)$ denote first-order partial derivatives in the x and y directions. If the matrix in the middle of Equation (37) is replaced with $\mathbf{M}$, Equation (37) has the quadratic form of an ellipse.

$$E(x,y) = \begin{bmatrix} x & y \end{bmatrix} \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix} \tag{38}$$

.

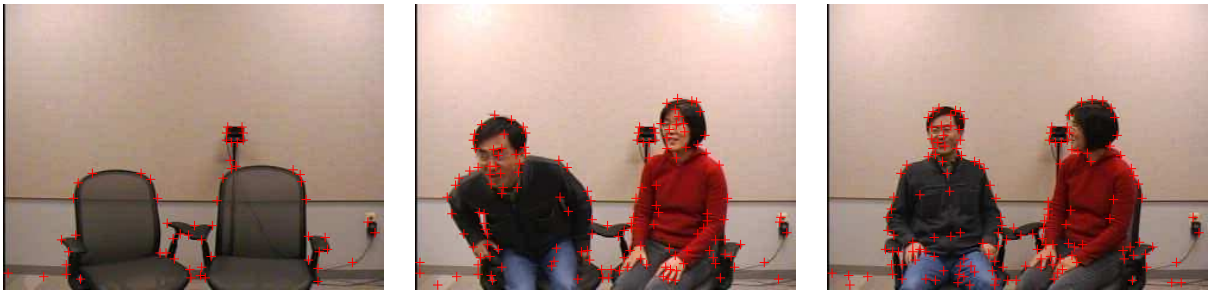
In the quadratic equation of an ellipse, the eigenvalues of $\mathbf{M}$ decide the shape of the ellipse. Since corners have high intensity changes in both principal directions, they have two large eigenvalues, while edges have a large eigenvalue and a small eigenvalue, and flat areas have two small eigenvalues. This indicates that the analysis of the eigenvalues of $\mathbf{M}$ is enough to distinguish corners from edges and flat areas. Each pixel is determined by evaluating a corner response, R, which is described using a determinant and a trace with two eigenvalues as

$$R = Det(M) - k(Trace(M))^2 \tag{39}$$

$$= (\lambda_1 \lambda_2) - k(\lambda_1 + \lambda_2)^2 \tag{40}$$

where λ_1 and λ_2 are two eigenvalues, and k is determined empirically. If R at a pixel is bigger than a threshold, the pixel is determined to be a corner. Figure 3.15 demonstrates the results of applying the above corner detection to our test sequence. The red crosses show the locations of detected corners. As in Figure 3.15, one-directional edges, for example, the bottom line of the wall panel, are not detected as

corners.



Frame 1 Frame 40 Frame 66

Figure 3.15: One result of the corner detection applied to our test sequence.

After corners have been extracted in each image, the next step is to find the best matching points between the corners of two consecutive images. This is done by calculating the cross-correlation over the current frame at each detected corner belonging to each object in the previous frame. However, after detecting the best matching corners, some matching corners are totally incorrect matches. These outliers should be deleted to get the best motion vector from the true matching corners belonging to each object. This deletion can be done using a median or random sample consensus (RANSAC). This research uses a median because it is simple. Figure 3.16 demonstrates the objects detected with motion vectors using corner detection/matching.

The frames used in Figure 3.16 are the same as the frames in Figure 3.14 applied for the mean-shift algorithm. The black rectangles show the position of the objects in the previous frame over the current frame, and the red rectangles show the position of the detected objects in the current frame. The red arrows connect two matching corners between two consecutive images. When one object does not have motion, the red arrows appear as points. At frames 83, 84, and 85, there are several outliers whose motion vectors are very different from the actual global motion vector for the object. The global motion vector for each object is obtained by deriving the median from all matching corners from the object.

The final results are relatively accurate regardless of whether the motion that is

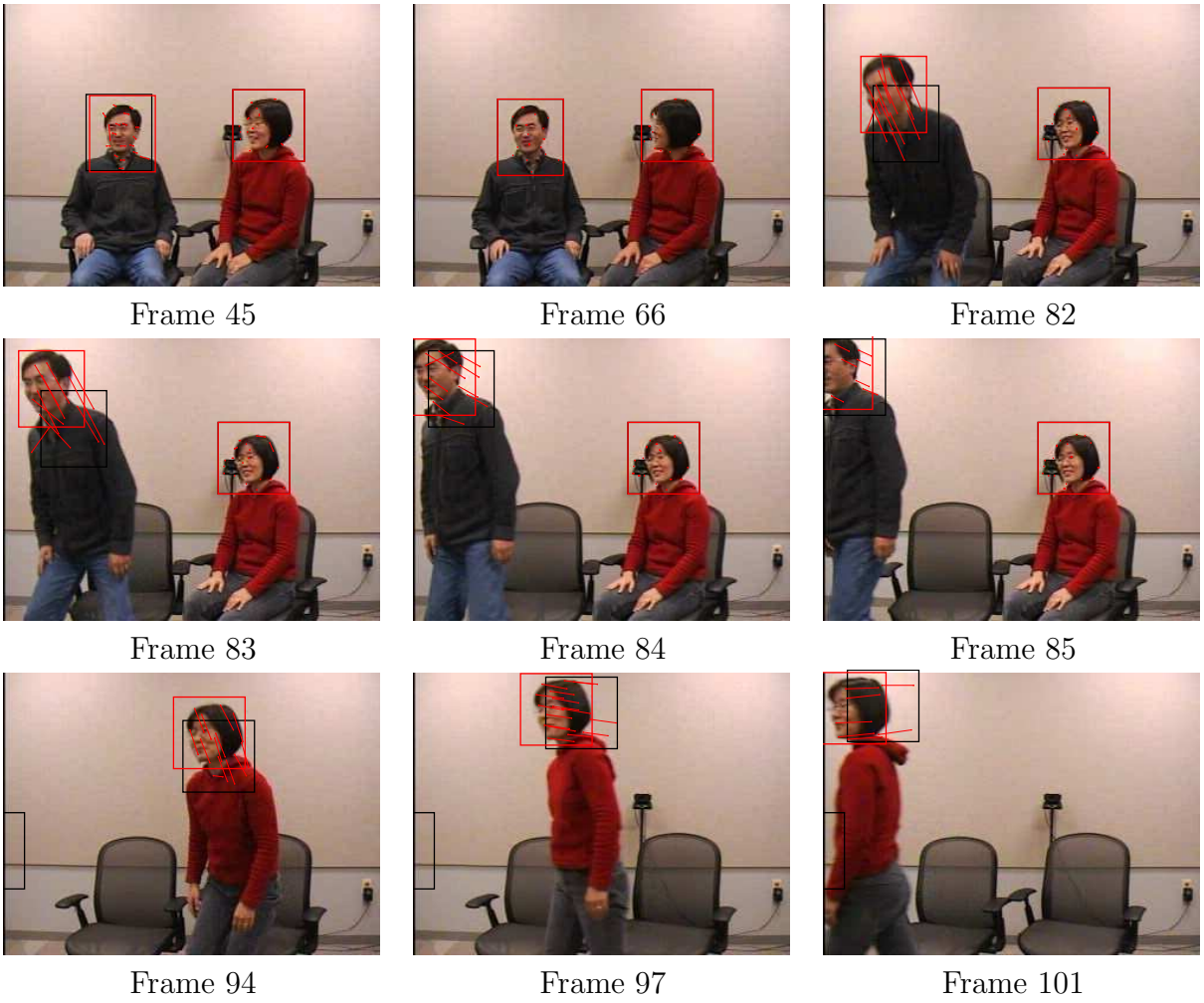


Figure 3.16: One result of motion detection using corner detection/matching.

occurring is fast or not, except the case when the object size becomes suddenly larger when an object approaches the camera. Then, the window, which is applied to the object, cannot contain the boundary of the face. The large face also shows all details of the face, so it generates many new corners. As a result, the number of matching corners suddenly decreases, and the motion vector detected from this small number of the matching corners is inaccurate.

3.4.5 Point-to-line Correspondence in Multiple Camera Geometry

The initialization with three Viola-Jones detectors and then motion detection using corner matching works very well for images having frontal faces/profiles like Figure 3.16, but it still cannot find sufficient features from the camera located behind the people. If the initial measurements are chosen as only two overlapping features as described in section 3.4.2, in order to use more reliable measurements, the upper body features detected from the back of the people cannot be used. Since the initialization step to estimate initial positions of the objects in particle filtering needs accurate measurements, the measurements detected from a single classifier cannot be used for the initialization because they can be inaccurate or completely false. However, they can be used for tracking because the data association, which will be described in Chapter 4, can solve the problem of falsely detected data. Therefore, this section describes correspondences between multiple cameras and proposes a simple method to decide which measurement can be made more robust by using the correspondence between multiple cameras.

In object detection using multiple cameras, using dependence between calibrated cameras is highly recommended because the geometrical dependence can help to derive better and more accurate measurements of objects. The most fundamental dependence is epipolar geometry, which is a point-to-line correspondence between two cameras described by a fundamental matrix described more fully in Appendix B. The relationship described by the fundamental matrix is between two cameras, and it can be extended to more than two cameras. Figure 3.17 shows this correspondence among three cameras.

The first row of Figure 3.17 shows a hand-picked point in camera 1 and the two corresponding lines in camera 2 and camera 3, which intersect with the correct person. The second row of Figure 3.17 also shows correct point-to-line correspondences

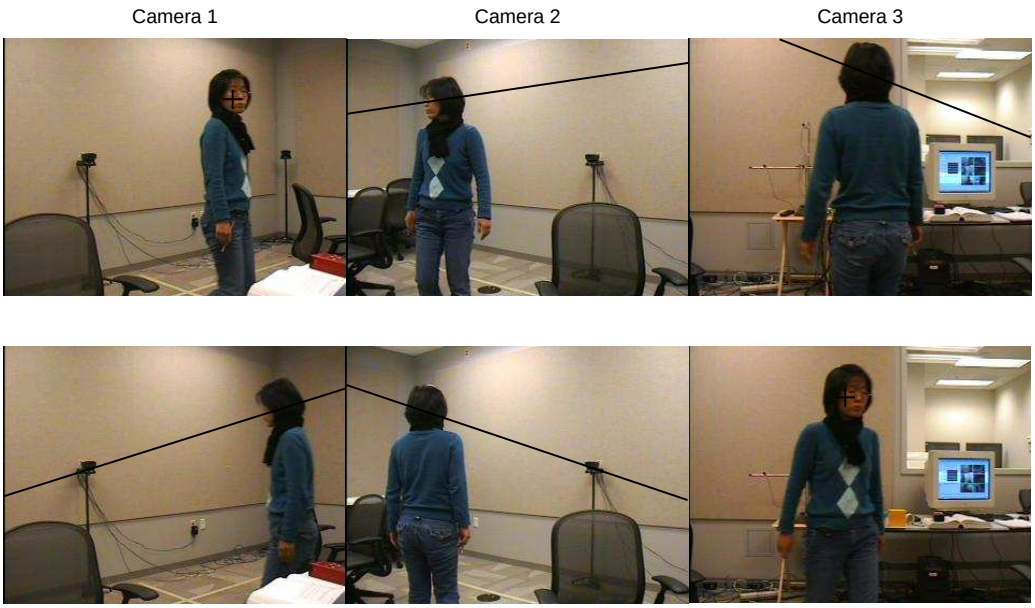


Figure 3.17: The point-to-line correspondence using the fundamental matrices of three cameras. A point in one camera is mapping into the lines of the other cameras.

between multiple cameras when a point in camera 3 is given. This point-to-line correspondence is independent of the scene and depends only on the geometry and internal characteristics of the cameras. Using the correspondence narrows the possible area where the undetected object is located in another camera, from a 2D whole image to a line.

When we have more than two cameras, and if any two cameras detect the same object, the point-to-line correspondences even further narrow the possible locations of objects in the third camera to a single point. In Figure 3.18, camera 1 detects a correct upper body of one object, and camera 2 also detects a correct face profile of the object, but camera 3 does not detect any feature of the object. However, when the point-to-line correspondences of two cameras are applied to camera 3, the location of the object is localized using the intersection of the two lines. However, this is an ideal case where we have only a single object, which is detected accurately in at least two cameras.

If there are falsely detected measurements, this point-to-line correspondence gives

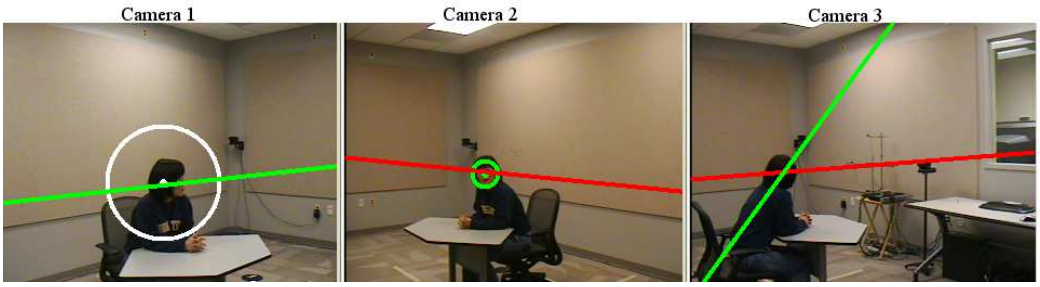


Figure 3.18: The results the fundamental matrix is applied to our images. Two cameras detected accurate position of the object, and the line correspondences driven from these points localize the correct position of the object in the third camera.

the wrong idea about the location of the object. Figure 3.19 shows a case that has a falsely detected measurement in a single object. The feature point detected from camera 1 in Figure 3.19 derives red lines in the other cameras, which cross the correct object. However, the object detected in camera 3 does not exist in camera 1 and camera 2, but blue lines are derived on these two cameras. Therefore, the intersection of two derived lines in camera 2 is the wrong position of the object.

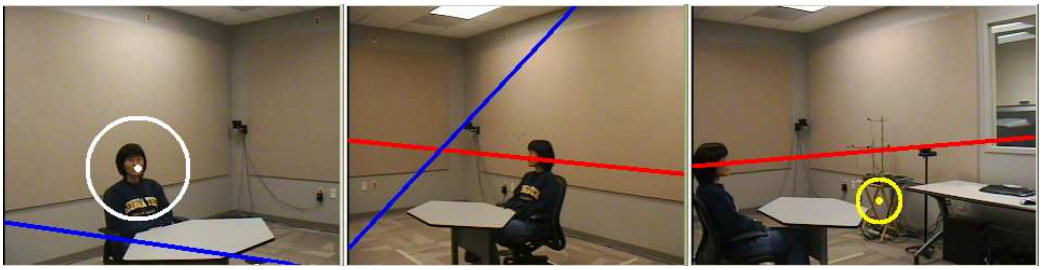


Figure 3.19: Only one camera detected accurate position of the object, and another camera has a wrong measurement. Then, the line correspondences driven from these points give a wrong position of the object in the third camera.

In the case of multiple objects, the relationship between line correspondences becomes more complicated even though there are only accurately detected objects as in Figure 3.20. In Figure 3.20, camera 1 detects one upper body and one frontal face. The two lines driven to camera 2 from the two features in camera 1 are nearly parallel and are close each other. The two lines driven to camera 3 from the same points in camera 1 are almost coincident because one object blocks the other. Therefore, the

leftmost object in camera 2 has two intersections, but it is hard to distinguish which intersection is correct. It is risky to detect new locations using intersections of two crossing lines since our Viola-Jones detector has a high falsely detected measurement rate. Therefore, this research does not use any intersection derived from corresponding lines, but only uses lines to determine whether a visual feature detected from single classifier is robust or not.

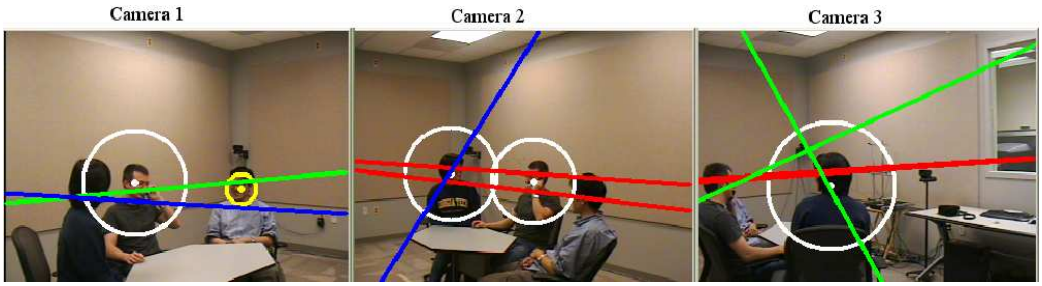


Figure 3.20: The point-line correspondences in case of multiple objects

The algorithm to determine robust visual features from detected features from only a single classifier uses a distance measure calculated from a point and a line, measured in pixels. First, all corresponding lines are derived from all measurements from Viola-Jones detectors using three features. Next, distances are calculated between the derived lines and robust feature points of the other cameras. If a distance is less than few pixels, the original visual feature matched for the line is classified as a robust feature.

The whole algorithm for visual feature detection is summarized in Figure 3.21. First, Viola-Jones detectors using three classifiers search of every frame of the whole image. At the same time, motion detection using corner detection are applied if there are any robust features detected from the previous frame. If there is a feature detected from two classifiers at the same time, it is compared with the current robust features from motion detection using corners, whether it is the same location or not. If it is a totally new feature, then it is added to the list of robust features. If it is the same as

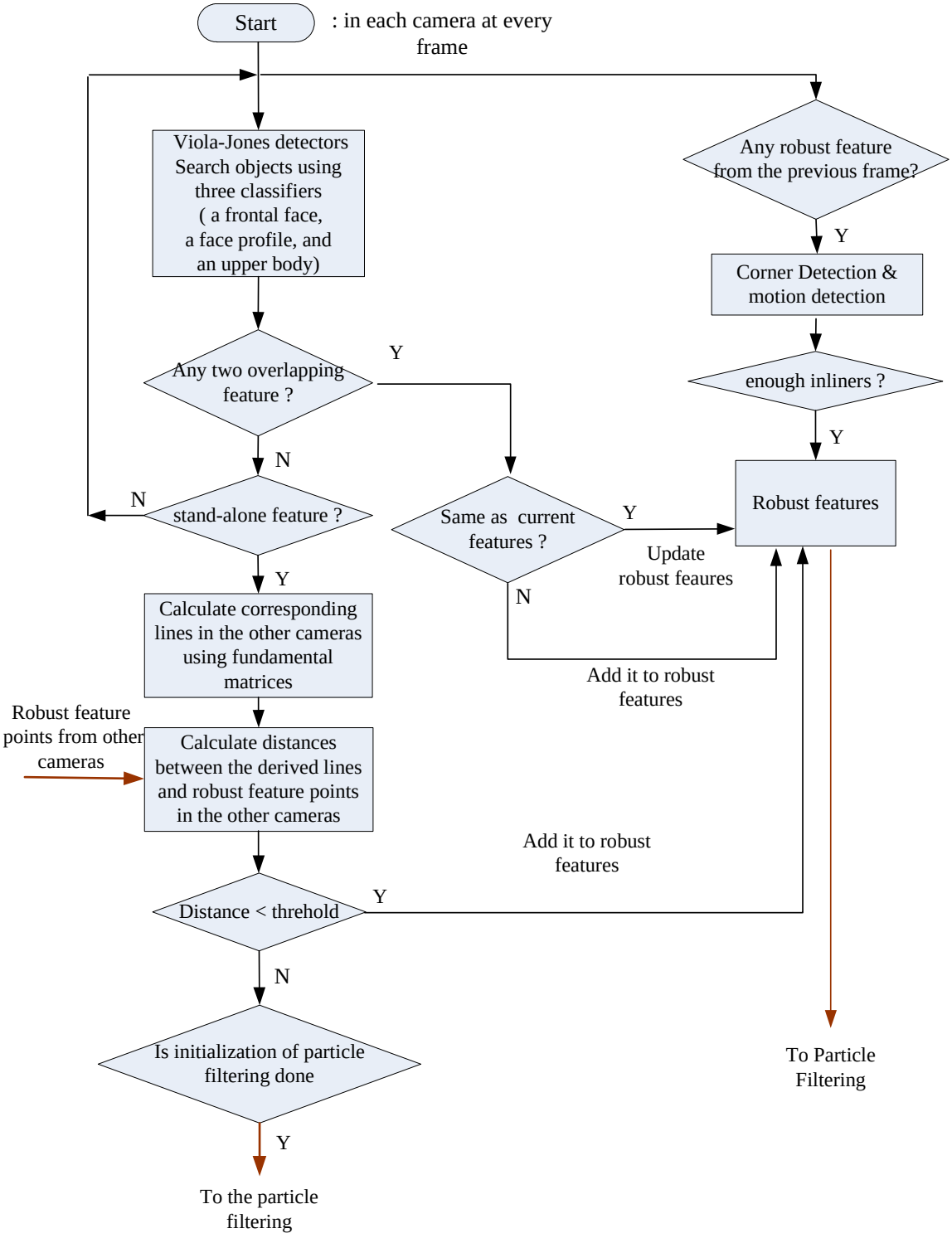


Figure 3.21: The whole visual feature detection algorithm

the previous robust feature, it updates the size of the current robust feature list. The detected features from only a single classifier is used to derive epipolar lines for the other cameras, and then the distances between the derived lines and robust feature points in the other cameras are calculated. If the distance is a few pixels, then the feature used for deriving the corresponding line is also classified as a robust feature, and then used for motion detection using corner detection. Then, the all features detected from only a single classifier, but not determined as robust features, are sent only when the particle filtering finishes its initialization and is working in a tracking mode. The procedure described in Figure 3.21 is executed in all cameras every frame.

3.5 Simulation for Visual Feature Detection

3.5.1 The Results of Face/head Detection using the Corner Matching

First, two cameras, which have been calibrated off-line, are installed as shown in Figure 3.22. Videos can be captured and stored. For the videos for this simulation, one person moves irregularly inside the view but sometimes goes out of the view. The face of the person is relatively small. The image size is 320 x 240, and the frame rate is 5 [frames/sec]. The total number of frames for each video is 100 frames.

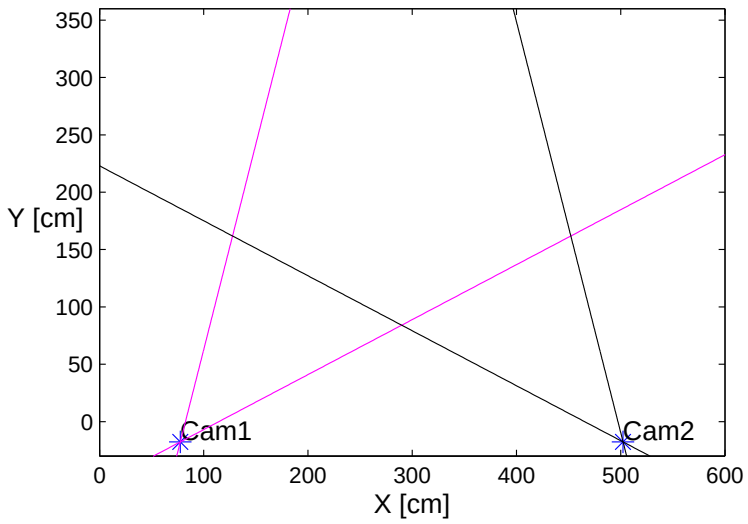
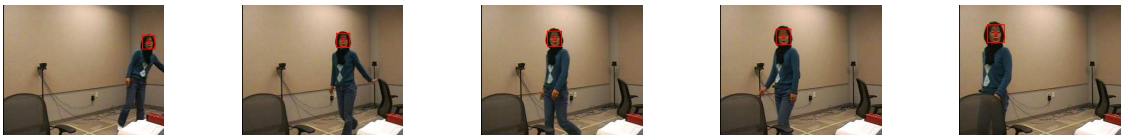


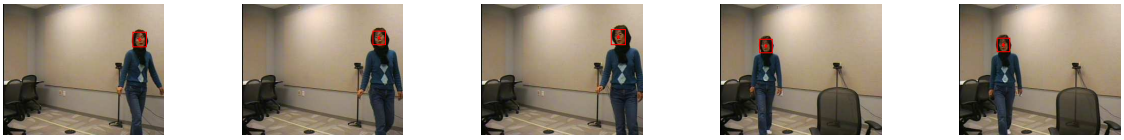
Figure 3.22: The positions of two cameras installed in a lab.

At initialization, the initial position of the object in the image is estimated using the three Viola-Jones detectors. The object is initialized when two different combinations of two of the three features are detected at the same time, for example, a frontal face and an upper body feature, a face profile and an upper body feature, or a frontal face and a face profile. As a result, the object is first detected at frame 30 and several times more from camera 1, as shown in Figure 3.23. From camera 2, the object is first detected at frame 21 and again later, as shown in Figure 3.24.



Frame 30 Frame 42 Frame 45 Frame 46 Frame 49

Figure 3.23: The positions of the object detected from two overlapping features in camera 1.



Frame 21 Frame 23 Frame 24 Frame 56 Frame 57

Figure 3.24: The positions of the object detected from two overlapping features in camera 2.

Corner detection occurs when the position of the object is first detected, as in Figure 3.23 and Figure 3.24. Since the window from the initialization includes only a face part, the window size for the corner detection/matching is magnified 1.6 times and now includes the outside boundary of the face. Figure 3.25 demonstrates the final results. The first and third columns show the results by applying the three Viola-Jones detectors to every frame. The second and fourth columns show the results by using the motion from corner detection/matching. These second and fourth columns are overlapping images of a current frame and the previous frame in order to give a clear idea of the true motion between the consecutive images. A red box indicates

the object in a current frame, and a black box represents the object in the previous frame.

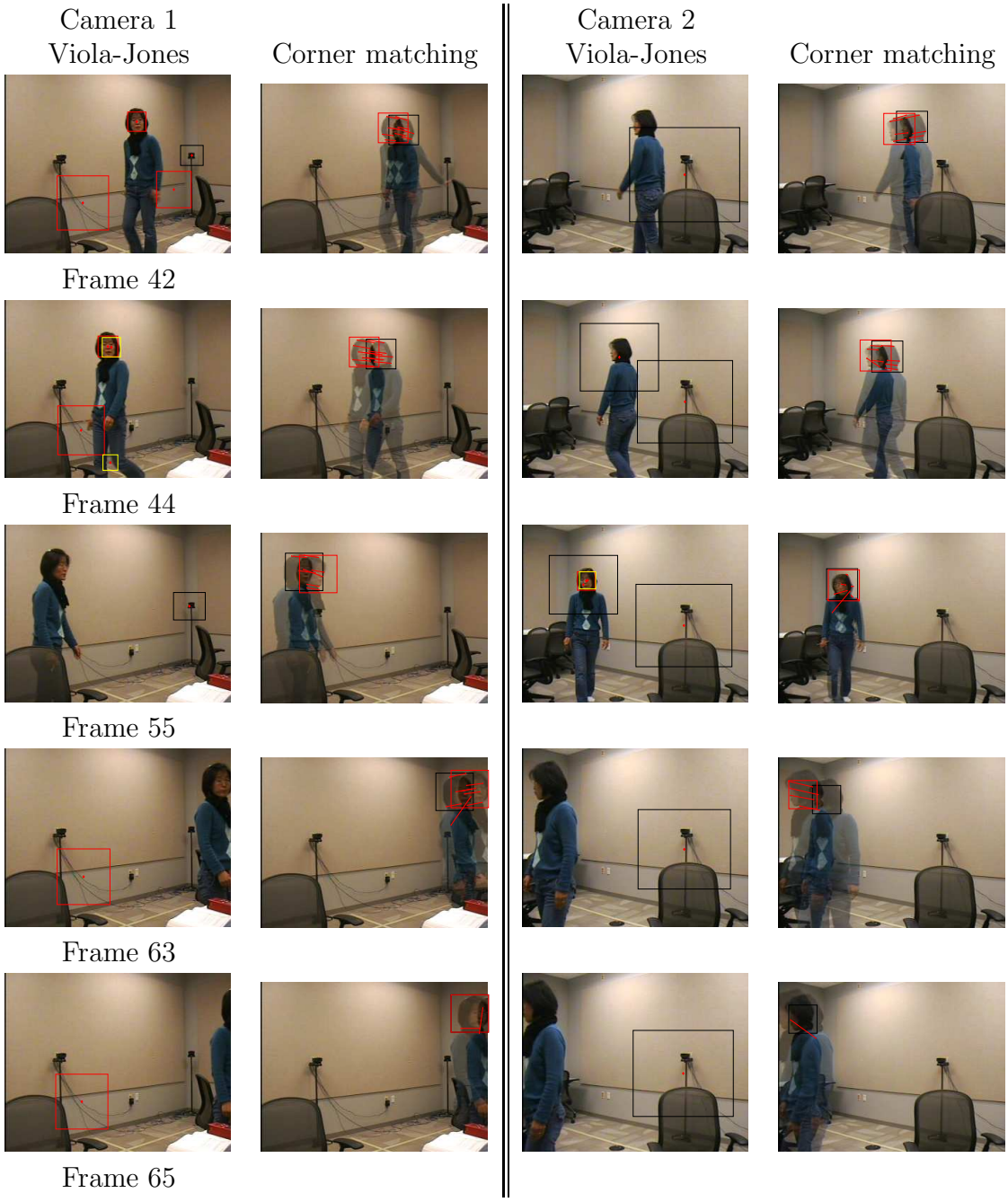


Figure 3.25: The results of object detection using motion vectors as a result of corner detection/matching compared with using the three features of the Viola-Jones detector.

Object detection using motion vectors from corner detection/matching works better than object detection with the Viola-Jones detectors except when the object in

the camera size becomes suddenly larger as a result of a change in object direction. When the object approaches the camera, the size of the object, in this case the face, can become larger than the window size used in the previous image, as in frame 65 in camera 2. The corners belonging to the object are not included in the window, and the motion vector becomes inaccurate.

3.5.2 The Results Applying Multiple Camera Geometry

Figure 3.27 shows robust feature detection using the algorithm in Figure 3.22. The four cameras were installed at each corner of the room as Figure 3.26, but the simulation in this section uses only three cameras.

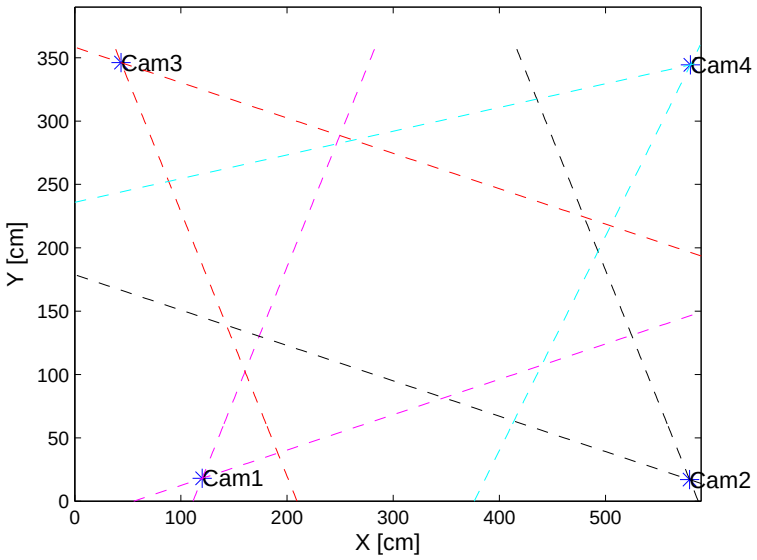
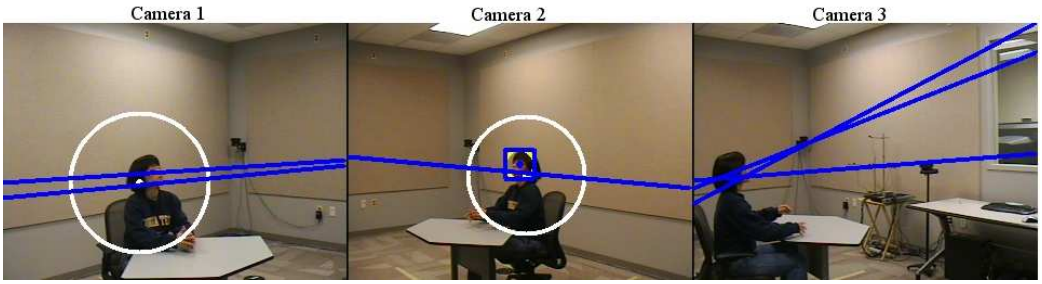


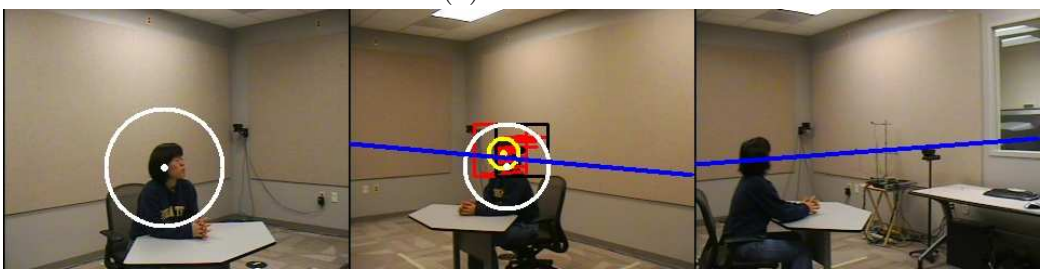
Figure 3.26: New positions of four cameras. Different from two cameras in Figure 3.22, these cameras were installed at the corners of the room.

In Figure 3.27, camera 2 finds one overlapping feature at frame 1, so it is tracked by motion detection using corner detection from frame 2 and determined as a robust feature. camera 1 finds one upper body feature at frame 2. The derived line to camera 2 from the upper body feature exactly crossed the robust feature in camera 2. Therefore, this upper body feature in camera 1 is also determined as a robust feature

at frame 3. At frame 6, camera 3 finds one upper body feature, and both derived lines from this point to camera 1 and camera 2 crossed robust features in each camera, so the upper body feature in camera 3 at frame 6 is also determined as a robust feature. As a result, all cameras detected right features at frame 7 and tracks the feature using motion estimation using corner detection/matching. Meanwhile, to detect a new object, Viola-Johns detector also search every frame.

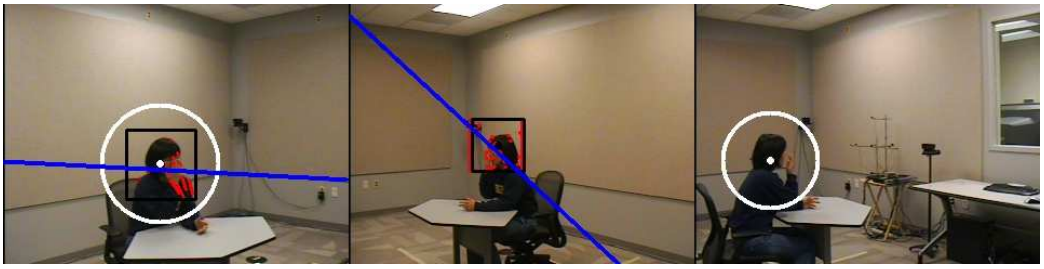


(a) Frame 1



(b) Frame 2

(c) Frame 3



(d) Frame 6

(e) Frame 7

Figure 3.27: Robust feature detection using selection of two overlapping features and point-to-line correspondences

CHAPTER IV

PARTICLE FILTERING FOR DATA ASSOCIATION AND SENSOR FUSION

4.1 Overview

As Chapter 2 addressed the need for data association for tracking in a real environment, this chapter introduces two approaches for data association in the framework of a particle filter [68]. The first method is a Monte-Carlo implementation of a joint probability data association filter (JPDAF). For data association for multiple targets, JPDAF generates data association hypotheses for all targets but processes each target separately, which results in the advantage that it does not suffer from the curse of dimensionality from multiple targets. However, one disadvantage of JPDAF is its computational complexity due to the increased number of data association hypotheses.

The other approach is data association with IPPF referred to here as DA-IPPF. This approach extends the independent partitioned particle filter for data association because the IPPF works very well with a small number of particles in multiple-target tracking when the data association is known. Data association using IPPF was tried by Vermaak [68], but our algorithm calculates the final weights differently using posterior probabilities and improves its algorithm by proposing an exclusive data association parameter for each target that shows the relationship between a measurement and a target. However, the assumption of independent sampling for the data association parameter in the IPPF framework contradicts the basic idea of data association that seeks to clarify the dependence between measurements and targets. Therefore, DA-IPPF can suffer from serious performance degradation when

the measurements for multiple targets are very close to each other because of its assumption of independence. Unlike JPDAF, DA-IPPF does not generate all possible data association hypotheses, which results in the advantage of reduced computational complexity.

These two approaches are derived using multiple sensors to implement sensor fusion as a part of the data association. Both solve missing data and falsely detected data difficulties in their frameworks through association hypotheses. However, since both approaches cannot handle the appearance of a new target, and the determination of the total number of targets are detected outside the data association.

This chapter introduces MC-JPDAF first followed by DA-IPPF. Then it also proposes an initialization method, which is critical to most tracking algorithms using importance sampling.

4.2 Data Association and Sensor fusion Based on a Bayesian Approach

4.2.1 Parameters for Data Association

Since this research assumes that there are multiple sensors, the notations for symbols used in Chapter 2 are modified to accommodate multiple sensors. N^o is the number of sensors. A superscript i always indicates sensor i or something related to sensor i. $\mathbf{z}_t$ includes all measurements from N^o sensors at time t, so $\mathbf{z}_t^i$ describes only measurements from sensor i among $\mathbf{z}_t$.

The problem of data association is that the association between targets and measurements is unknown. Therefore, all feasible hypotheses need to be considered as candidates for the association. A target-to-measurement association hypothesis is denoted by a joint state with $\lambda^i = (\mathbf{r}^i, M_C^i, M_T^i)$ for sensor i. Given M^i measurements from sensor i and K targets at time t, M_C^i is the amount of clutter, which is the number of measurements not belonging to targets, M_T^i is the number of detected targets, and $\mathbf{r}^i = (r_1^i \dots r_k^i \dots r_K^i)$ is an association vector. r_k^i is a measurement number for

target k such as

$$r_k^i = \begin{cases} 0 & \text{if target k is not detected at sensor i} \\ j \in \{1 \dots M^i\} & \text{if target k generates measurement j at sensor i.} \end{cases}$$

As an example, if it is assumed that there are two measurements from one sensor for two targets, the superscript i for sensor number is not used because there is only one sensor. Then, $K = 2$, $M = 2$, and $\mathbf{r} = \{r_1, r_2\}$, whose value is 0 or one of the measurements such as

$$r_1 = 0, 1, \text{or } 2,$$

$$r_2 = 0, 1, \text{or } 2.$$

Then, the total number of possible association hypotheses is seven, as shown in Table 4.5. Assigning the same measurement to multiple targets is not allowed. Therefore, $(r_1 = 1, r_2 = 1)$ and $(r_1 = 2, r_2 = 2)$ are deleted from the association hypotheses.

Table 4.5: An example of data association hypotheses for JPDAF using $K = 2$, $M = 2$, and $N^o = 1$.

hypothesis	h1	h2	h3	h4	h5	h6	h7
r_1	0	1	0	2	0	1	2
r_2	0	0	1	0	2	2	1
M_C	2	1	1	1	1	0	0
M_T	0	1	1	1	1	2	2

Next, the prior probability for a data association hypothesis and the data likelihood conditioned by this data association hypothesis are derived. The prior probability for a data association hypothesis, $p(\lambda_t^i)$, is a joint probability of $\mathbf{r}^i$, M_C^i, and M_T^i. Using the chain rule, this joint probability factors into three terms. Since the number of measurements for clutter is independent of the number of detected targets,

the prior probability is simplified to Equation (41).

$$\begin{aligned} p(\lambda_t^i) &= p(\mathbf{r}^i, M_C^i, M_T^i) \\ &= p(\mathbf{r}^i | M_C^i, M_T^i) p(M_C^i | M_T^i) p(M_T^i) \\ &= p(\mathbf{r}^i | M_C^i, M_T^i) p(M_C^i) p(M_T^i) \end{aligned} \tag{41}$$

The first term, $p(\mathbf{r}^i | M_C^i, M_T^i)$ is a uniform distribution of the number of all possible association hypotheses given M_C^i and M_T^i as

$$p(\mathbf{r}^i | M_C^i, M_T^i) = \frac{1}{C(K, M_T^i) P(M^i, M_T^i)}. \tag{42}$$

Given K, M_T^i, and M_C^i, the number of association hypotheses is the permutation of M_T^i elements from M^i elements (i.e., $P(M^i, M_T^i) = \frac{M^i!}{M^i - M_T^i!}$) multiplied by the combination of M_T^i elements from K elements (i.e., $C(K, M_T^i) = \binom{K}{M_T^i}$). M_T^i is always less than or equal to K feasible hypotheses.

The second term in Equation (41), $p(M_C^i)$, is the probability of clutter and is usually modeled as a Poisson probability as

$$p(M_C^i) = \lambda^{M_c^i} \frac{\exp(-\lambda)}{M_c^i!}. \tag{43}$$

The last term in Equation (41), $p(M_T^i)$, is the probability of the detected targets, which can be modeled as a binomial distribution of M_t^i elements that are present among K targets with a detection rate of P_D as

$$p(M_T^i) = \binom{K}{M_T^i} P_D^{M_T^i} (1 - P_D)^{K - M_T^i}. \tag{44}$$

After combining all three terms, the prior probability for a data association hypothesis becomes

$$p(\lambda_t^i) = (\frac{M_T^i!}{M^i!}) \lambda^{M_c^i} \frac{\exp(-\lambda)}{M_c^i!} (1 - P_D)^{K - M_T^i} P_D^{M_T^i}, \tag{45}$$

which will be used to calculate the posterior probability of a target-to-measurement hypothesis.

The data likelihood conditioned by the data association hypothesis, $p(\mathbf{z}_t^i|\mathbf{x}_t, \lambda_t^i)$ adds a target-to-measurement hypothesis to $p(\mathbf{z}_t^i|\mathbf{x}_t)$. If it is assumed that all measurements are independent of each other given a state and a data association hypothesis, the conditional data likelihood is described as in Equation (46). Then, it is divided into two parts according to the values of r_k^i, as in Equation (47).

$$
\begin{aligned}
p(\mathbf{z}_t^i|\mathbf{x}_t, \lambda_t^i) &= p(z_{1,t}^i, \ldots, z_{M^i,t}^i|\mathbf{x}_t, \lambda_t^i) \\
&= \prod_{j=1}^{M^i} p(z_{j,t}^i|\mathbf{x}_t, \lambda_t^i) && (46) \\
&= \prod_{j\in\{r_k^i=0\}} p(z_{j,t}^i) \prod_{j\in\{r_k^i=j, j\neq 0\}} p(z_{j,t}^i|\mathbf{x}_{k,t}) && (47) \\
&= \prod_{j\in\{r_k^i=0\}} p_C(z_{j,t}^i) \prod_{j\in\{r_k^i=j, j\neq 0\}} p(z_{j,t}^i|\mathbf{x}_{k,t}) && (48) \\
&= V^{i-M_C^i} \prod_{j\in\{r_k^i=j, j\neq 0\}} p(z_{j,t}^i|\mathbf{x}_{k,t}) && (49)
\end{aligned}
$$

The first product in Equation (47) is the probability of clutter, denoted by p_C in Equation (48). This is usually a uniform probability over a measurement space of sensor i, i.e., V^i. The second product in Equation (47), the data likelihood for a detected target is denoted by $p(z_{j,t}^i|\mathbf{x}_{k,t})$, which needs to be defined from the data likelihood model for each target k.

4.2.2 Gating

In an environment of high clutter, it is important to remove all inaccurate measurements to reduce the number of data association hypotheses. The most common procedure used to validate measurements is gating. Gating sets up a validation region or a gate around a predictive measurement. The predictive measurement is calculated for each target and each sensor as

$$
p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1}) = \int p_k(\mathbf{z}_t^i|\mathbf{x}_{k,t}) p(\mathbf{x}_{k,t}|\mathbf{Z}_{t-1}) d\mathbf{x}_{k,t}. \tag{50}
$$

To draw a validation region using $p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1})$ on the measurement space of sensor i, the probability of Equation (50) should be approximated by a probability that can associated by a smooth region, such as a Gaussian probability. Therefore, $p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1})$ is approximated as

$$p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1}) \sim N(\hat{\mathbf{z}}_t^i, \Sigma_{\mathbf{z}^i}) \tag{51}$$

where $\hat{\mathbf{z}}_t^i$ and $\Sigma_{\mathbf{z}^i}$ are the mean and the variance of the predictive measurement of sensor i for target k. Then, the validation region, denoted by $\hat{V}^i$, is found as

$$\hat{V}^i \triangleq \{(\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i)^T \Sigma_{\mathbf{z}^i}^{-1} (\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i) \leq \gamma\}. \tag{52}$$

The shape of the validation region depends on the dimension of the measurement space. A line is used for one-dimensional space, an ellipse for two-dimensional spaces, etc. The size or volume of $\hat{V}^i$ depends on γ, which is determined from a χ^2 distribution with the number of degrees of freedom of the measurement space and a detection rate. If a measurement falls in the validation region, it is determined to be a valid measurement.

4.2.3 Joint Probability Data Association Filter (JPDAF)

This section introduces an algorithm for data association, a joint PDAF (JPDAF), which is extended from PDAF for multiple targets. The posterior probability in JPDAF is almost the same as it is in Equation (5), only calculated for target k:

$$p_k(\mathbf{x}_{k,t}|\mathbf{Z}_t) = \frac{p(\mathbf{x}_{k,t}|\mathbf{Z}_{t-1}) p_k(\mathbf{z}_t|\mathbf{x}_{k,t})}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})} \tag{53}$$

To implement Equation (53), the filtering for each target is done using a predefined $p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1})$; however, the updating cannot be done directly because the association between $\mathbf{z}_t$ and $\mathbf{x}_{k,t}$ is unknown. Therefore, our effort in JPDAF focuses on how to define and calculate this data likelihood, $p(\mathbf{z}_t|\mathbf{x}_{k,t})$. JPDAF calculates the data likelihood by retaining all feasible measurements and reflecting their target-to-measurement contributions in the data likelihood. A new parameter, β_{jk}^i is defined

as a posterior probability for the data association of target k to measurement j from sensor i. Using β_{jk}^i, the data likelihood is described by

$$\begin{aligned} p_k(\mathbf{z}_t|\mathbf{x}_{k,t}) &= \prod_{i=1}^{N^o} p_k(\mathbf{z}_t^i|\mathbf{x}_{k,t}) \\ &= \prod_{i=1}^{N^o} \big(\beta_{0k}^i + \sum_{j=1}^{M^i} \beta_{jk}^i p_k(z_{j,t}^i|\mathbf{x}_{k,t})\big). \end{aligned} \tag{54}$$

β_{0k}^i is the probability that there is no measurement from sensor i for target k. Since β_{jk}^i is the posterior probability of a target-to-measurement association as $\beta_{jk}^i = p_k(r_k^i = j|\mathbf{Z}_t)$, β_{jk}^i is a summation of all association hypotheses having $r_k^i = j$ over the set of valid joint target-to-measurement association hypotheses as

$$\beta_{jk}^i = p_k(r_k^i = j|\mathbf{Z}_t) = \sum_{\{\lambda_t^i \in \Lambda_t^i, r_k = j\}} p(\lambda_t^i|\mathbf{Z}_t) \tag{55}$$

where Λ_t^i is the sample space of all feasible data association hypotheses.

The $p(\lambda_t^i|\mathbf{Z}_t)$ in Equation (55) is

$$\begin{aligned} p(\lambda_t^i|\mathbf{Z}_t) &\propto p(\lambda_t^i) p(\mathbf{z}_t^i|\lambda_t^i, \mathbf{Z}_{t-1}) \\ &\propto p(\lambda_t^i)(V^i)^{-M_c^i} \prod_{k=1}^{K} p_{r_k^i=j}(z_t^i = j|\mathbf{Z}_{t-1}). \end{aligned} \tag{56}$$

The $p_{r_k^i}(z_{j,t}^i|\mathbf{Z}_{t-1})$ in Equation (56) is calculated using Equation (8) as

$$p_{r_k^i}(z_t^i = j|\mathbf{Z}_{t-1}) = \int p_k(z_t^i = j|\mathbf{x}_{k,t}) p(\mathbf{x}_{k,t}|\mathbf{Z}_{t-1}) d\mathbf{x}_{k,t} \tag{57}$$

where $p(\mathbf{x}_{k,t}|\mathbf{Z}_{t-1})$ is the result from the filtering of Equation (6). Then, β_{jk}^i in Equation (55) is finally calculated from the summation over all data association hypotheses having $r_k^i = j$ using the prior probability of Equation (45) and the predictive measurement of Equation (57).

The calculation of β_{jk}^i is also demonstrated using the same example that we used in Table 4.5. For example, $\beta_{1,1}$ for $j = 1$ and $k = 1$ is the summation of hypothesis

h_2 and h_6 in Table 4.5 as

$$\begin{aligned}\beta_{11} &= p(\lambda_t = h2|\mathbf{Z}_t) + p(\lambda_t = h6|\mathbf{Z}_t) \\ &= p(\lambda_t)(V)^{-1} p_{r_1=1}(z_t = 1|\mathbf{Z}_{t-1}) + p(\lambda_t) p_{r_1=1}(z_t = 1|\mathbf{Z}_{t-1}) p_{r_2=2}(z_t = 2|\mathbf{Z}_{t-1}) \\ &= p(\lambda_t)(V)^{-1} p_{k=1}(z_{1,t}|\mathbf{Z}_{t-1}) + p(\lambda_t) p_{k=1}(z_{1,t}|\mathbf{Z}_{t-1}) p_{k=2}(z_{2,t}|\mathbf{Z}_{t-1}).\end{aligned}$$

4.3 Monte-Carlo Joint Probability Data Association Filter (MC-JPDAF)

The JPDAF explained so far implements an analytic, linear probability density not a non-linear density. Since this research uses the framework of a particle filter, which implements a non-linear, non-Gaussian system with discrete samples, a data association method should work within the framework of the particle filter. Therefore, JPDAF is implemented using a Monte-Carlo method, called MC-JPDAF. In MC-JPDAF, each step described in JPDAF is implemented in a Monte-Carlo method. The major difference between MC-JPDAF and a particle filter is that MC-JPDAF does not calculate a joint posterior probability for multiple targets, $p(\mathbf{x}_t|\mathbf{Z}_t)$. Instead, MC-JPDAF calculates only a posterior probability for each target, $p_k(\mathbf{x}_{k,t}|\mathbf{Z}_t)$. A Monte-Carlo implementation of the gating will be explained, and a proposal function for the importance sampling will be proposed. The whole algorithm of MC-JPDAF is summarized in Table 4.6.

4.3.1 Monte-Carlo Implementation of the Gating

The gating explained in Section 4.2.2 is implemented with a Monte-Carlo method. To calculate the predictive measurement for each sensor of Equation (50), two inner parts of the integral in the equation have to be implemented in a Monte-Carlo method. The first one is a predictive state as

$$p(\mathbf{x}_{k,t}^{(n)}|\mathbf{Z}_{t-1}) = p(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)}) p(\mathbf{x}_{k,t-1}^{(n)}|\mathbf{Z}_{t-1}). \tag{58}$$

This is implemented using importance sampling. When new particles at time t are generated using a proposal function of $q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)$, the $p(\mathbf{x}_{k,t}^{(n)}|\mathbf{Z}_{t-1})$ denoted by $\alpha_{k,t}^{(n)}$ is approximated with

$$\alpha_{k,t}^{(n)} \propto w_{k,t-1}^{(n)} \frac{p_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)})}{q_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)}. \tag{59}$$

Using $\alpha_{k,t}^{(n)}$, the predictive measurement of Equation (50) using the Monte-Carlo implementation is

$$p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1}) \approx \sum_{n=1}^{N} \alpha_{k,t}^{(n)} p(\mathbf{z}_t^i|\mathbf{x}_{k,t}^{i(n)}). \tag{60}$$

The data likelihood for each sensor in Equation (60), $p(\mathbf{z}_t^i|\mathbf{x}_{k,t}^{(n)})$, is approximated with a Gaussian distribution as $p(\mathbf{z}_t^i|\mathbf{x}_{k,t}^{(n)}) = N(\mathbf{z}_t^i|\hat{\mathbf{z}}(\mathbf{x}_{k,t}^{i(n)}), \Sigma_{\mathbf{z}_t^i})$, where $\hat{\mathbf{z}}_t(\mathbf{x}_{k,t}^{i(n)})$ is calculated from the correspondence between $\mathbf{x}_{k,t}^{(n)}$ and the measurement space of sensor i. The probability of the predictive measurement is finally approximated as a single Gaussian distribution in order to draw a validation region below.

$$\begin{aligned} p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1}) &\approx \sum_{n=1}^{N} \alpha_{k,t}^{(n)} N(\mathbf{z}_t|\hat{\mathbf{z}}(\mathbf{x}_{k,t}^{i(n)}), \Sigma_{\mathbf{z}_t^i}) \\ &\approx N(\mu_{k,t}^i, \Sigma_{k,t}^i) \end{aligned} \tag{61}$$

The mean, $\mu_{k,t}^i$, and the variance, $\Sigma_{k,t}^i$, are approximated for each target and each sensor as

$$\mu_{k,t}^i = \sum_{n=1}^{N} \alpha_{k,t}^{(n)} \hat{\mathbf{z}}(\mathbf{x}_{k,t}^{i(n)}) \tag{62}$$

$$\Sigma_{k,t}^i = \sum_{n=1}^{N} \alpha_{k,t}^{(n)} (\hat{\mathbf{z}}(\mathbf{x}_{k,t}^{i(n)}) - \mu_{k,t}^i)^T (\hat{\mathbf{z}}(\mathbf{x}_{k,t}^{i(n)}) - \mu_{k,t}^i) \tag{63}$$

The validation region for measurement of sensor i is as

$$\hat{V}^i \triangleq \{(\mathbf{z}_t^i - \mu_{k,t}^i)^T {\Sigma_{k,t}^i}^{-1} (\mathbf{z}_t^i - \mu_{k,t}^i) \leq \gamma\}. \tag{64}$$

4.3.2 Proposal Function

A well-designed proposal function can greatly improve the efficiency of generated particles, but simplicity of implementation should be also considered. This section suggests two proposal functions. The simplest proposal function is a state update model, which is used most frequently in the literature of the particle filter such as a bootstrap filter [20], as

$$q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t) \sim p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}). \tag{65}$$

The other one uses a mixture model such as

$$q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t) \sim r^D p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}) + (1 - r^D) \sum_{i=1}^{N^o} \sum_{j=1}^{M^i} \gamma_j^i p_k^i(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}) p(\mathbf{z}_{j,t}^i|\mathbf{x}_{k,t}). \tag{66}$$

Here, r^D and γ_j^i are weights for the Gaussian mixture. The mixture model performs better than the state update model because it generates particles in all possible areas that may have targets, but it requires a lot of particles.

Table 4.6 summarizes the MC-JPDAF step by step.

4.4 Particle Filter with Data Association (DA-IPPF)

While MC-JPDAF is just a MC implementation of JPDAF, the particle Filter with data association derives a data association method from the posterior probability of the particle filter [68].

Let us start with a posterior probability, $p(\mathbf{x}_t|\mathbf{Z}_t)$. Since it is impossible to get this posterior probability with unknown data association, the target state is augmented by a data association hypothesis, λ_t, as

$$p(\mathbf{x}_t, \lambda_t|\mathbf{Z}_t) = p(\lambda_t) p(\mathbf{z}_t|\mathbf{x}_t, \lambda_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1}) d\mathbf{x}_{t-1} \tag{67}$$

The integrand in Equation (67) is not a function of λ_t, and the prior for a data association hypothesis, $p(\lambda_t)$, and the data likelihood conditioned on a data association

Table 4.6: Summary of MC-JPDAF

For k=1…K,
  let us assume we have $\{w_{k,t-1}^{(n)}, \mathbf{x}_{k,t-1}^{(n)}\}_{n=1}^{N}$.
For k=1…K, n=1…N,
  generate new samples for the target states $\mathbf{x}_{k,t}^{(n)} \sim q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)$.
For $k = 1 \dots K, n = 1 \dots N$,
  computer and normalize $p(\mathbf{x}_{k,t}|\mathbf{Z}_{t-1})$:
  $\alpha_{k,t}^{(n)} \propto w_{k,t-1}^{(n)} \frac{p_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)})}{q_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)}$
For $k = 1 \dots K, i = 1 \dots N^o$
  compute the predictive measurement:
  $p_k(\mathbf{z}_t^i|\mathbf{Z}_{t-1}) \approx N(\mu_{k,t}^i, \Sigma_{k,t}^i)$
Gating: Choose a proper γ using a χ^2 distribution
  and delete non-feasible measurements.

For $i = 1 \dots N^o$,
  generate joint target-to-measurement association hypotheses at sensor i,
  $\Lambda_t^i = \{\lambda_t^i\}$.
For $i = 1 \dots N^o, \lambda_t^i \in \Lambda_t^i$,
  computer the joint association posterior probability of Equation (56).
For $k = 1 \dots K, i = 1 \dots N^o, j = 0 \dots M^i$,
  compute a marginal association posterior probability of Equation (55),
  β_{jk}^i.
For $k = 1 \dots K, n = 1 \dots N$
  compute a data likelihood for each target, $p_k(\mathbf{z}_t|\mathbf{x}_{k,t}^{(n)})$ in Equation (54):
  $p_k(\mathbf{z}_t|\mathbf{x}_{k,t}^{(n)}) = \prod_{i=1}^{N^o} \left(\beta_{0k}^i + \sum_{j=1}^{M^i} \beta_{jk}^i p_k(z_{j,t}^i|\mathbf{x}_{k,t}^{(n)})\right)$.
For $k = 1 \dots K, n = 1 \dots N$,
  computer and normalize the particle weights:
  $w_{k,t}^{(n)} \propto w_{k,t-1}^{(n)} \frac{p_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)}) p_k(\mathbf{z}_t|\mathbf{x}_{k,t}^{(n)})}{q_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t}^{(n)}, \mathbf{z}_t)}$.
For $k = 1 \dots K, n = 1 \dots N$,
  if resampling is required, resample it.

hypothesis, $p(\mathbf{z}_t|\mathbf{x}_t, \lambda_t)$ have been already derived in Equation (45) and Equation (49), so they are simply to be extended for multiple sensors as

$$p(\lambda_t) = \prod_{i=1}^{N^o} p(\lambda_t^i) \tag{68}$$

$$p(\mathbf{z}_t|\mathbf{x}_t, \lambda_t) = \prod_{i=1}^{N^o} \left(V^{i-M_C^i} \prod_{k=1}^{K} p_k(\mathbf{z}_{r_k^i=j,t}^i|\mathbf{x}_{k,t})\right). \tag{69}$$

The derivation of the particle filter for data association is already done.

The straight forward implementation of this particle filter with data association is summarized in Table 4.7. Both $\mathbf{x}_t^{(n)}$ and $\lambda_t^{(n)}$ are generated, given $\mathbf{x}_{t-1}^{(n)}$ and $\lambda_{t-1}^{(n)}$. However, when the posterior probability of $p(\mathbf{x}_{t-1}|\mathbf{z}_{t-1})$ is known, $\lambda_{t-1}^{(n)}$ does not affect $\lambda_t^{(n)}$. Therefore, $\lambda_t^{(n)}$ is temporally independent. $\mathbf{x}_t^{(n)}$ and $\lambda_t^{(n)}$ are generated using a proposal function as

$$(\lambda_t^{(n)}, \mathbf{x}_t^{(n)}) \sim q(\lambda_t, \mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t). \tag{70}$$

Since both $\mathbf{x}_t^{(n)}$ and $\lambda_t^{(n)}$ cannot be generated at the same time, the proposal function is partitioned as

$$\begin{aligned} q(\lambda_t, \mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) &= q(\lambda_t|\mathbf{x}_t, \mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) \\ &= q(\lambda_t|\mathbf{x}_t, \mathbf{z}_t) q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t). \end{aligned} \tag{71}$$

Particles of $\mathbf{x}_t^{(n)}$ are first generated using $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t)$, and then particles for $\lambda_t^{(n)}$ are generated using $q(\lambda_t|\mathbf{x}_t^{(n)}, \mathbf{z}_t)$. In the multiple-sensor environment, $\lambda_t^{i(n)}$ is generated as

$$\begin{aligned} \mathbf{x}_t^{(n)} &\sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) \\ \lambda_t^{i(n)} &\sim q(\lambda_t^i|\mathbf{x}_t^{(n)}, \mathbf{z}_t). \end{aligned}$$

This whole algorithm for data association in Table 4.7 works well for single-target tracking, but suffers from the curse of dimensionality because a large number of particles is needed to maintain acceptable performance. For this reason, IPPF, explained in Chapter 2 is approached as a base framework and extended with data association, shown in Table 4.8, called DA-IPPF.

First, the $\mathbf{x}_{k,t}^{(n)}$ are generated. The generated $\mathbf{x}_{k,t}^{(n)}$ are used for selecting a data association hypothesis. For generating an association hypothesis, an association parameter is generated instead of λ_t^i because $M_c^{i(n)}$ and $M_T^{i(n)}$ are determined by the association parameter $\mathbf{r}_{k,t}^{i(n)}$.

In the framework of MC-JPDAF, $r_{k,t}^{i(n)}$ should be $r_{k,t}^{i(n)} \neq \{j \in r_{1:k-1,t}^{i(n)}\}$. However, having the same $r_{k,t}^{i(n)}$ for multiple targets can be possible in the DA-IPPF because

Table 4.7: Extension of a standard particle filter with data association.

For n= $1 \dots N$,
  generate particles for $\mathbf{x}_t^{(n)}$:
$\mathbf{x}_t^{(n)} \propto q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t)$.
For $i = 1 \dots N^o$, n= $1 \dots N$,
  generate particles for λ_t^i:
  $\lambda_t^{i(n)} \propto q(\lambda_t^i|\mathbf{x}_t^{(n)}, \mathbf{z}_t)$.
For n= $1 \dots N$,
  compute and normalize the particle weights:
  $w_t^{(n)} \approx w_{t-1}^{(n)} \frac{p(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}) \prod_{i=1}^{N^o} p(\lambda_t^i) \prod_{i=1}^{N^o} p(\mathbf{z}_t^i|\mathbf{x}_t^{(n)}, \lambda_t^{(n)})}{q(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) \prod_{i=1}^{N^o} \left(q(\lambda_t^{i(n)}|\mathbf{x}_t^{(n)}, \mathbf{z}_t^i)\right)}$.
  $\sum_{n=1}^{N} w_t^{(n)} = 1$
For $i = 1 \dots N$,
  if resampling is required, resample it.

Table 4.8: Summary of DA-IPPF.

For $k = 1 \dots K, n = 1 \dots N$,
  generate particles for target states.
  $\mathbf{x}_{k,t}^{(n)} \propto q(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)$
For $k = 1 \dots K, i = 1 \dots N^o, n = 1 \dots N$,
  generate particles for a target-to-measurement association parameter.
  $r_{k,t}^{i(n)} \propto q(r_{k,t}^i|, \mathbf{x}_{k,t}^{(n)}, \mathbf{z}_t^i)$
For $k = 1 \dots K, n = 1 \dots N$,
  compute partitioned target weights.
  $\beta_{k,t}^{(n)} \propto \frac{p_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)})}{q_k(\mathbf{x}_{k,t}^{(n)}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t)} \prod_{i=1}^{N^o} \frac{p(\mathbf{z}_{r_{k,t}^{i(n)}}|\mathbf{x}_{k,t}^{(n)}) p(r_{k,t}^{i(n)})}{q(r_{k,t}^{i(n)}|\mathbf{x}_{k,t}^{(n)}, \mathbf{z}_t^i)}$
For $k = 1 \dots K$,
  normalize partitioned target weights.
  $\sum_{n=1}^{N} \beta_{k,t}^{(n)} = 1$
For $k = 1 \dots K, i = 1 \dots N^o, , j = 1 \dots N$,
  resample $\{\mathbf{x}_{k,t}^{(j)}, \{r_{k,t}^{i,(j)}\}$ and based on $\{\beta_{k,t}^{(n)}\}_{n=1}^{N}$ and replace them.
For $n = 1 \dots N$,
  compute and normalize particle weights:
  $w_t^{(n)} \approx w_{t-1}^{(n)} \frac{p(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}) \prod_{i=1}^{N^o} \left(p(\lambda_t^{i(n)}) p(\mathbf{z}_t^i|\mathbf{x}_t^{(n)}, \lambda_t^{i(n)})\right)}{q(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t) \prod_{i=1}^{N^o} q(\lambda_t^{i(n)}|\mathbf{x}_t^{(n)}, \mathbf{z}_t^i)}$.
For $n = 1 \dots N$,
  if resampling is required, sample it.

the IPPF is constructed based on independence assumption between multiple targets. This independence between multiple targets indicates that $r_{k,t}^{i(n)}$ is also assumed to

be independent among multiple targets. As a result of this assumption, the same measurement can be assigned to different targets. This is more likely to happen when two targets are located very close to each other or when the measurements of multiple targets are close. Assigning the same measurement to multiple targets has one advantage when two targets cross because a true measurement for either target can be lost, for example, by occlusion. However, it cannot be a solution for general occlusion because assigning the same measurement to multiple targets may lead to continuously assigning the same measurement to both targets even after the two targets become separated. Therefore, DA-IPPF prohibits assigning the same measurement into multiple targets. Since the DA-IPPF does not calculate all possible hypotheses as MC-JPDAF does, a simple exclusive generation method of association parameters is proposed.

First, $r_{k,t}^{i(n)}$ is generated independently for each target. Then, duplication of $r_{k,t}^{i(n)}$ among multiple targets is tested for each particle. If the same association parameter (except $r_{k,t}^{i(n)} = 0$) is assigned to more than one target, the data likelihood conditioned on the same data association parameter is calculated for the targets. $r_{k,t}^{i(n)}$ is kept for the target that has the highest data likelihood, while $r_{k,t}^{i(n)}$ is regenerated for the other targets, excluding the previous parameter, which is already assigned.

Even though the generation of the same $r_{k,t}^{i(n)}$ is not allowed, the resampling process for each target generates the same $r_{k,t}^{i(n)}$ again because resampling is a process that copies particles having high weights without considering data association.

4.4.1 Proposal Function

DA-IPPF needs two proposal functions to generate $\mathbf{x}_{k,t}^{(n)}$ and $r_{k,t}^{i(n)}$. As shown for the standard particle filter with data association, the joint proposal function factors into two terms in Equation (71). $\mathbf{x}_{k,t}^{(n)}$ is generated using either Equation (65) or Equation (66). Then, a data association parameter is generated for each target and

sensor using a proposal function as

$$q(r^i_{k,t} = j|\mathbf{x}^{(n)}_{k,t}, \mathbf{z}^i_t) = \frac{p(\mathbf{z}^i_t|r^i_{k,t} = j, \mathbf{x}^{(n)}_{k,t})p(r^i_{k,t} = j|\mathbf{x}^{(n)}_{k,t})}{\sum_{l=0}^{M^i} p(\mathbf{z}^i_t|r^i_{k,t} = l, \mathbf{x}^{(n)}_{k,t})p(r^i_{k,t} = l|\mathbf{x}^{(n)}_{k,t})}$$
$$= \frac{p(\mathbf{z}^i_t|r^i_{k,t} = j, \mathbf{x}^{(n)}_{k,t})p(r^i_{k,t} = j)}{\sum_{l=0}^{M^i} p(\mathbf{z}^i_{l,t}|r^i_{k,t} = l, \mathbf{x}^{(n)}_{k,t})p(r^i_{k,t} = l)}. \quad (72)$$

$p(\mathbf{z}^i_t|r^i_{k,t} = j, \mathbf{x}^{(n)}_{k,t})$ is

$$p(\mathbf{z}^i_t|r^i_{k,t} = j, \mathbf{x}^{(n)}_{k,t}) = \begin{cases} V^{i-M^i} & \text{if j=0} \\ V^{i-M^i+1} p(z_{j,t}|\mathbf{x}^{(n)}_{k,t}) & \text{otherwise.} \end{cases} \quad (73)$$

The prior $p(r^i_{k,t})$ is simply a detection rate if $r_{k,t}$ is not 0 as

$$p(r^i_{k,t} = j) = \begin{cases} 1 - p_D & \text{if } r^i_{k,t} = 0 \\ \frac{p_D}{K} & \text{otherwise.} \end{cases} \quad (74)$$

Because the same association parameter is not allowed, the prior is not correct but suffices.

4.5 Initialization

Before tracking, the number of targets and the initial target positions need to be determined. Even though most of the literature did this initialization manually or assumed to know them, this is not likely in the real environment. This section proposes an initialization method to find initial target positions using multiple measurement association hypotheses and the maximum data likelihood. The initial idea for this initialization method came from the target-to-measurement association hypotheses in JPDAF.

Again, there are N^o sensors. The measurements from sensor i at $t = 0$ is $\mathbf{z}^i_0 = \{z^i_{j,t=0} : j = 1, \ldots, M^i\}$. $z^i_{j,0}$ has only two possibilities: a true measurement or a falsely detected measurement. Then, the total number of possible associations of measurements from N^o sensors, N_h, is a product of the number of measurements from

each sensor plus one for missing data. Here, it should be clear that this association is not a target-to-measurement association in JPDAF, but just association between measurements, so we call it measurement association here.

$$N_h = \prod_{i=1}^{N^o} (M^i + 1). \tag{75}$$

For example, assuming that there are three sensors and each sensor has two measurements, the number of possible associations is $(2+1)^3 = 27$ and the association space of measurements is $\Omega = \{s_j : j = 1, \ldots, 27\}$. Each measurement hypothesis can have a true measurement association for at most one target.

Table 4.9: One example of measurement association hypotheses for initialization with $N^o = 3$, $M^1 = 2$, $M^2 = 2$, and $M^3 = 2$.

	sensor1	sensor2	sensor 3
s_1	0	0	0
s_2	0	0	z_1^3
s_3	0	z_1^2	0
s_4	z_1^1	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_{26}	z_1^1	z_2^2	z_2^3
s_{27}	z_2^1	z_2^2	z_2^3

The data likelihood given a measurement association hypothesis is defined similar to Equation (49) as

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{x}_t, s_{j,t}) &= p(z^1_{s(1),t}, \ldots, z^{N^o}_{s(N^o),t}|\mathbf{x}_t, s_{j,t}) \\ &= \prod_{j=1}^{N^o} p(z^j_{s(j),t}|\mathbf{x}_t, s_{j,t}) \\ &= \prod_{z^j_{s(j),t}=0} p_C(z^j_{s(j),t}) \prod_{z^j_{s(j),t}\neq 0} p(z^j_{s(j),t}|\mathbf{x}_t) \\ &= \prod_{z^j_{s(j),t}=0} V^{j-1} \prod_{z^i_{s(j),t}\neq 0} p(z^j_{s(j),t}|\mathbf{x}_t) \end{aligned} \tag{76}$$

V^i is also a measurement space for sensor i.

Then, to find the right measurement association for unknown targets, the maximum values of the data likelihoods of all measurement association hypotheses are

compared. Since each hypothesis includes at most one true measurement association, the number of targets is estimated as the number of hypotheses whose data likelihood is greater than a threshold. The initial positions are the state that has the maximum data likelihood. Table 4.10 summarizes the proposed initialization procedure.

Table 4.10: Initialization of multiple targets using importance sampling.

For n=1 ... N,
generate particles over a possible tracking area using an uniform distribution: $q(\mathbf{x}_t|\mathbf{z}_t) \sim U(0,1)$
Generate measurement association hypotheses, $\Omega = \{s_j : j = 1, \ldots, N_h\}$.
For $i = 1 \ldots N_h, n = 1 \ldots N$,
compute a data likelihood based on each measurement association hypothesis $p(\mathbf{z}_t|\mathbf{x}_t^{(n)}, s_i)$ using in Equation (76).
For $j = 1 \ldots N_h$,
find a maximum value for each hypothesis using $p_{s_j,max} = p(\mathbf{z}_t|\mathbf{x}_t^{(n)}, s_j)$, find the estimated number of targets where the maximum data likelihood for each hypothesis is greater than a pre-defined threshold(ζ_0) as $\hat{K} \triangleq Count(p_{s_j,max} > \zeta_0)$
For $k = 1 \ldots \hat{K}$
estimate the states $\hat{\mathbf{x}}_k = max_{\mathbf{x}_t} p(\mathbf{z}_t|\mathbf{x}_t^{(n)}, s_j)$

4.6 Discussion

This chapter introduced two methods of data association for multiple-target tracking using multiple sensors, based on particle implementation. The first method was MC-JPDAF based on the Bayesian JPDAF. It is a Monte-Carlo implementation of JPDAF and avoids the curse of dimensionality, but the computational complexity increases greatly if the environment has a high clutter rate when the number of targets is greater than 2. For example, when the number of targets is 3, if there are three measurements from one sensor, the number of target-to-measurement association hypotheses for the sensor is $P(4,3) = \frac{4!}{1!} + 1 = 25$. However, if the number of measurements is increased one more by clutter, then the number of target-to-measurement association hypotheses is increased to $P(5,3) = \frac{5!}{2!} + 1 = 61$. If there are two additional clutter measurements, the number of data association hypotheses becomes $P(6,3) = \frac{6!}{3!} + 1 =$

120. These target-to-measurement association hypotheses should be generated for each sensor.

On the other hand, DA-IPPF does not generate all possible hypotheses, so the calculation complexity is much less than with JPDAF. However, since a particle filter has much more computation than JPDAF because it generates a lot of particles, the entire calculation for DA-IPPF is still more than with MC-JPDAF.

JPDAF or MC-JPDAF usually needs gating to reduce the number of association hypotheses. Sometimes, there is a possibility that incorrect γ for the gating can delete feasible measurements or vice versa. However, DA-IPPF does not need gating because generating samples of data association hypotheses using Equation (73) does not select measurements that are far from the predicted state.

Both methods work well in tracking multiple targets except target-crossing or measurement-crossing. At target-crossing, a target-to-measurement parameter for each target is found to have the other's measurement. This is fine when real targets are close, but a problem occurs after target-crossing. When measurements are separated after target-crossing, the estimated states for each target should be separated, but the result is sometimes switched, which means the state for target 1 tracks target 2 after crossing in MC-JPDAF. DA-IPPF turns out to have more serious performance degradation in the same situation. Even though the same association parameter is prohibited during the generation of association parameters, the resampling process for each target repeats the same association parameter for different targets. As a result, each state for multiple targets follows the same target, resulting in losing one target completely.

DA-IPPF and MC-JPDAF are applied to acoustic source tracking, visual source tracking, and joint audio-visual target tracking in the following chapters. More details of the analysis and simulation results for these data association methods follow in the next three chapters.

CHAPTER V

PARTICLE FILTERING FOR MULTIPLE TARGET TRACKING USING MULTIPLE, SINGLE-TYPE SENSORS

5.1 Overview

This chapter is designed to apply the feature detection, data association, and initialization algorithms described in Chapter 3 and in Chapter 4 to two real tracking scenarios using sensors of the same type, such as multiple speaker tracking using microphones and people tracking using multiple cameras. The block diagram of speaker tracking scenario is presented in Figure 5.1.

N^m microphones capture and send delayed acoustic signals synchronously to the acoustic-feature detection, which estimates TDOAs from pairs of microphones. The TDOA measurements are detected from N^a microphone pairs, not from N^m sensors. The number of TDOA measurements can vary in time, depending on the acoustic environment, the estimated number of speakers, and the particular microphone pairs in use.

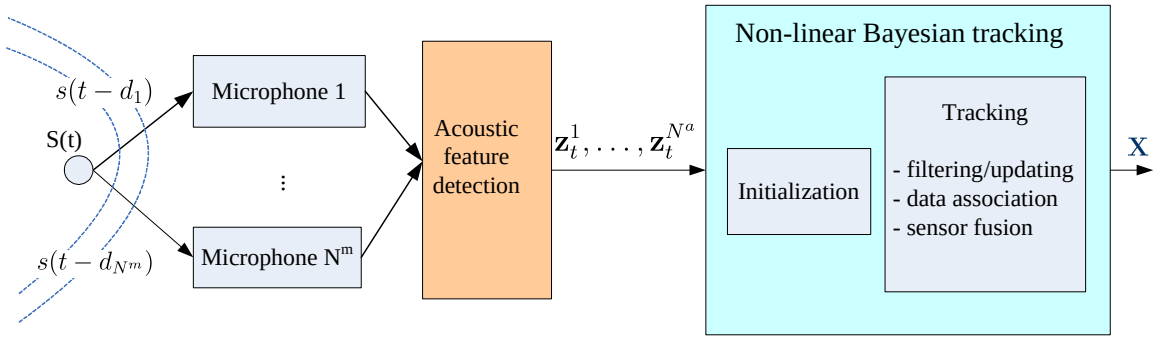


Figure 5.1: Block diagram of speaker tracking using particle filtering.

People tracking using multiple cameras is shown in Figure 5.2. The block diagram

is similar to the block diagram of the speaker tracking system except that in the diagram given here, visual-feature detection is performed independently for each camera image. While most past research for visual people tracking using particle filtering was done using 2D camera images [9, 54], the research presented here tracks 3D positions of people with multiple cameras.

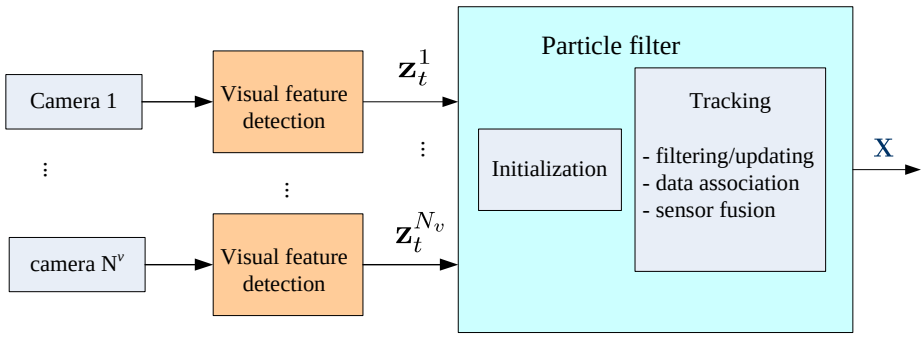


Figure 5.2: Block diagram of people tracking using particle filtering.

Either the TDOA measurements or the visual measurements then go to a particle filter, which contains the initialization and tracking components. While initialization finds the number of targets and the initial positions of the targets, tracking tracks the position of the targets in time using particle filtering.

This chapter proposes data models for particle filtering such as a state-space model, a state update model, and a data likelihood model for each tracking scenario. The problems implemented and evaluated in this chapter are the detection of multiple target locations at initialization, the tracking of both single and multiple targets, data association of unlabeled measurements, and sensor fusion from sensors of a single type.

5.2 Data Models

A particle filter is developed by deriving a state-space model, such as a state space, a state update model, and a data likelihood function. These data models are derived under the assumption that there are multiple targets and multiple sensors. All notation follows the notation used in Chapter 2 and Chapter 4.

5.2.1 A State Space and a State Update Model

A speaker in a multiple speaker tracking environment can be modeled as a point source. The location of the acoustic source, i.e., the person, is centered around the mouth. A person in multiple people tracking can also be modeled as a point source. The state then indicates the location of the center of the person's face. The state space for both is designed only with positions as

$$\mathbf{X}_t = [\mathbf{x}_{1,t} \dots \mathbf{x}_{k,t} \dots \mathbf{x}_{K,t}] \tag{77}$$

where $\mathbf{x}_{k,t} = [x_{k,t}, y_{k,t}, z_{k,t}]^T$ corresponds to x, y, and z, the positions of target k at time t in Cartesian coordinates. K is the number of targets.

Since a conference setting is our most likely application, the targets are assumed not to move significantly in a specific direction. Based on this assumption, the state update model is defined as a random walk in Equation (78). However, if the targets have distinct motion, the variance of the random walk model should cover the possible area of target motion. A different state update model can be applied to each target if each target has different state-transition dynamics, but here the same state update model is applied to all the targets.

$$p(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}) \sim \mathcal{N}(\mathbf{x}_{k,t-1}, \Sigma_{\mathbf{x}}) \tag{78}$$

5.2.2 Data Likelihood

5.2.2.1 Acoustic Data Likelihood

A TDOA is estimated using the PHAT from microphone pair i, consisting of microphones p and q. The TDOA is modeled using Equation (28) as

$$\hat{\mathbf{z}}_t^i = \lfloor \left(\hat{\tau}_{pq} F_s\right) \text{ [samples]}.$$

The data likelihood model for the measurements of microphone pair i, $p(\mathbf{z}_t^i|\mathbf{x}_t)$ is formulated using a Gaussian distribution:

$$p(\mathbf{z}_t^i|\mathbf{x}_t) \propto \exp\big[-\frac{1}{2}(\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i)^T \Sigma_a^{-i} (\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i)\big] \tag{79}$$

where $\mathbf{z}_t^i$ is a real measurement, and $\hat{\mathbf{z}}_t^i$ is an estimated measurement calculated with $\mathbf{x}_t$ and the locations of microphone pair i using Equation (28). Σ_a^{-i} is a diagonal matrix.

The data likelihood for each sensor pair is the same Gaussian distribution with a different variance because the variance changes according to the locations of each sensor pair. The total data likelihood is described as a product of the data likelihood of each sensor, as in

$$p(\mathbf{z}_t|\mathbf{x}_t) = \prod_{i=1}^{N^a} p(\mathbf{z}_t^i|\mathbf{x}_t) \tag{80}$$

5.2.2.2 Visual Data Likelihood

The visual measurements are extracted from the visual-feature detection procedure described in Section 3.4. Therefore, here, the measured locations of faces/heads and the width of a bounding box are assumed to be given. Measurement j from camera i is denoted by $\mathbf{z}_{j,t}^i = (u_{j,t}^i, v_{j,t}^i, w_{j,t}^i)$, where $(u_{j,t}^i, v_{j,t}^i)$ is the location of the center of face/head j and $w_{j,t}^i$ is the width of the bounding box containing face j from camera i.

The data likelihood function shows the relation between the measurement and its hidden state. The measurement consists of two parts of locations and sizes as $\mathbf{z}_t^i = (\mathbf{L}_t^i, \mathbf{w}_t^i)$. Accordingly, the data likelihood factors two components

$$p(\mathbf{z}_t^i|\mathbf{x}_t) = p(\mathbf{L}_t^i, \mathbf{w}_t^i|\mathbf{x}_t) \propto p(\mathbf{L}_t^i|\mathbf{x}_t)p(\mathbf{w}_t^i|\mathbf{x}_t). \tag{81}$$

The relationship between a position in an image, $\mathbf{L}_t^i$, and a point in 3D Cartesian coordinates, $\mathbf{x}_t$, is defined by a line using the camera-calibration matrix as

$$\hat{\mathbf{L}}_t^i = k\mathbf{P}^i\mathbf{x}_t \tag{82}$$

where $\mathbf{P}^i$ is the camera-calibration matrix for camera i. Therefore, the first term in Equation (81), the data likelihood for locations, $p(\mathbf{L}_t^i|\mathbf{x}_t)$, can be written in terms of

a multivariate Gaussian density as

$$p(\mathbf{L}_t^i|\mathbf{x}_t) \propto \exp\Big[-\frac{1}{2}\big[\mathbf{L}_t^i - \hat{\mathbf{L}}_t^i(\mathbf{x}_t, \mathbf{P}^i)\big]^T \Sigma_L^{i,-1}\big[\mathbf{L}_t^i - \hat{\mathbf{L}}_t^i(\mathbf{x}_t, \mathbf{P}^i)\big]\Big]. \tag{83}$$

The size of a face in a camera is inversely proportional to the distance between the camera and the face. Wu in [73] has inferred the distance between a camera and a head to be

$$d = f\frac{w_f}{w_i} \tag{84}$$

where d is the distance between the camera and the head, and w_f is the average width of a human head. w_i is the width of the bounding box, which contains the hair and skin parts of the head as they appear in the image. f is the horizontal focal length of the camera. The measurement, w_t^i, which is the width of the bounding box containing the hair and skin of the face, corresponds to $\mathbf{w}_i$ in Equation (84). Then, the relation between $\mathbf{w}_t^i$ and $\mathbf{x}_t = (x_t, y_t, z_t)$ is

$$\hat{\mathbf{w}}_t^i = f^i\frac{w_f}{d_t} = f^i\frac{w_f}{\sqrt{(x_i - x_t)^2 + (y_i - y_t)^2 + (z_i - z_t)^2}} \tag{85}$$

where (x_i, y_i, z_i) is the location of camera i, and f^i is the horizontal focal length of camera i, which is obtained from the camera-calibration matrix for camera i. The data likelihood for w_t^i for camera i is formulated using a multivariate Gaussian density as

$$p(\mathbf{w}_t^i|\mathbf{x}_t) \propto \exp\Big[-\frac{1}{2}\big[\mathbf{w}_t^i - \hat{\mathbf{w}}_t^i(\mathbf{x}_t, f^i, w_f)\big]^T \Sigma_{\mathbf{w}^i}^{-1}\big[\mathbf{w}_t^i - \hat{\mathbf{w}}_t^i(\mathbf{x}_t, f^i, w_f)\big]\Big]. \tag{86}$$

Using two different measurements improves the performance when one camera or a small number of cameras are used for tracking because the face location in the image gives the direction of the object and because the size of the face indicates the distance between the object and each camera, i.e., the range. Using both the direction and the range can improve tracking performance over using only the direction.

The data likelihood for the measurements from all cameras is the product of the data likelihood of each camera:

$$p(\mathbf{z}_t|\mathbf{x}_t) = \prod_{i=1}^{N^v} p(\mathbf{z}_t^i|\mathbf{x}_t). \tag{87}$$

5.2.3 Implementation of Sensor Fusion using Single-type Sensors

The sensor fusion is implemented using particles as in Figure 5.3 when there is no delay between sensors. The particles are generated outside the sensors, and then each sensor shares the same particles.

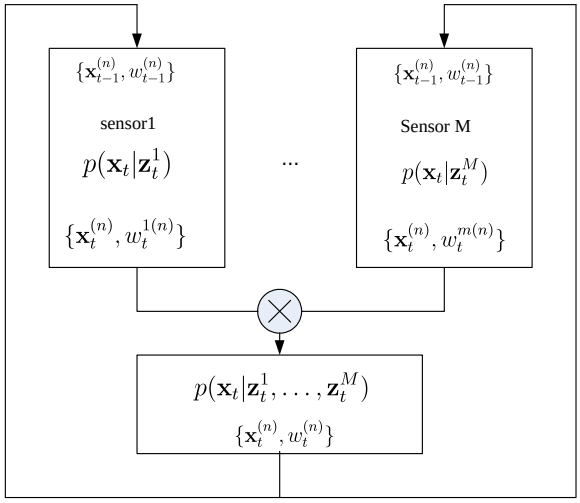


Figure 5.3: Implementation of sensor fusion using single-type sensors.

Equation (65) or Equation (66) can be used to generate these particles. The posterior probability from each sensor is implemented as particles. Then, the final joint posterior probability is a simple product of weights calculated from each sensor.

5.3 Simulation Results

5.3.1 Simulation Environment

The simulation is performed in the same room as the one described in Section 3.3. Six microphones are located according to the configuration in Figure 5.4. "M1", ..., and "M6" refer to the locations of the microphones. Unlike the simulation environment

described in Section 3.3, the recorded acoustic data are not used as simulation sources here. This is primarily because it is difficult to control the simulation conditions such as measurement errors, missing data rates, and falsely detected data rates by simply changing the signal-to-noise ratio. TDOAs are instead calculated using the positions of the sources and microphones.

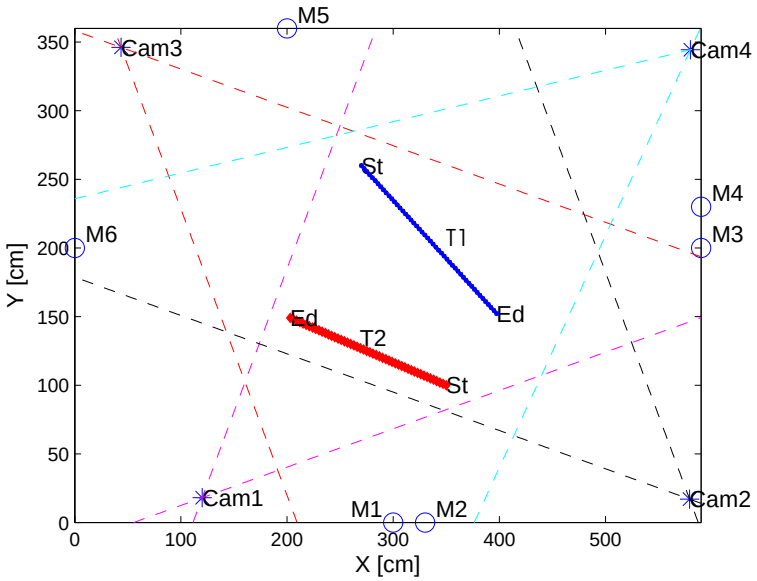


Figure 5.4: Trajectory of two people in the x-y plane.

For visual tracking, four cameras are installed around the room in Figure 5.4 and are aimed at the center of the room. Figure 5.4 shows the positions of the cameras and the horizontal angle of view of each camera in the x-y plane. All cameras are calibrated off-line [1]. The image size is 320 x 240 and the frame rate is 5 frame/sec. The number of cameras varies from one to four.

For a comparison of objective performance, a mean absolute error (MAE) is calculated for 20 Monte-Carlo repetitions

$$MAE = \frac{1}{20}\sum_{n=1}^{20} |\mathbf{x}_t(n) - \hat{\mathbf{x}}_t(n)| \; cm, \tag{88}$$

where $\mathbf{x}_t$ is a true position and $\hat{\mathbf{x}}_t$ an estimated position.

The particle filter uses the state update model as the proposal function. The parameters for particle filtering for both speaker tracking and visual tracking are summarized in Table 5.11.

Table 5.11: The parameters used in particle filtering.

$$\Sigma_{\mathbf{x}} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_x^2 \end{bmatrix} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

$$\Sigma_v = \begin{bmatrix} \sigma_l^2 & 0 & 0 \\ 0 & \sigma_l^2 & 0 \\ 0 & 0 & \sigma_w^2 \end{bmatrix} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

$$\Sigma_a = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$N = 500$

This simulation applies three different noise conditions: additive Gaussian measurement noise, missing data, and falsely detected data. For the first case, additive, zero-mean Gaussian measurement noise with a variance σ_z^2 is added to the measurements. The missing data are implemented by deleting correct measurements whenever a Poisson process for λ_m generates an error. Falsely detected data are generated with a uniform distribution over the measurement space whenever a Poisson process for λ_f generates an error. For example, $\lambda_m = 0.1$ indicates that 10% of the true data are missed. Likewise, $\lambda_f = 0.1$ indicates that 10% of the total data is incorrect data. V_a^i, the size of the measurement space for each microphone pair is two times larger than the maximum TDOA values in the microphone pair. The size of the measurement spaces for the visual data is the image size, as shown in Table 5.12.

Table 5.12: The size of the measurement space, V^i of acoustic and visual measurements.

Mic pair 1	V_a^1	162
Mic pair 2	V_a^2	338
Mic pair 3	V_a^3	384
Mic pair 4	V_a^4	358
Mic pair 5	V_a^5	368
Location of a Face	$V_{\mathbf{L}}$	320x240 (image size)
Size of a face	V_w	320 (horizontal image size)

5.3.2 Initialization

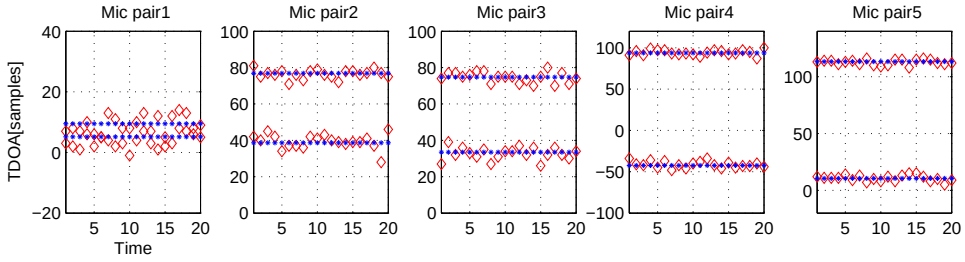
This section investigates the performance of the initialization algorithm proposed in Section 4.5. The targets are assumed to be fixed at the initial position of the trajectory at $(270, 260, 160)$ and $(350, 100, 170)$, as shown in Figure 5.4.

Initialization for the Acoustic Tracking :

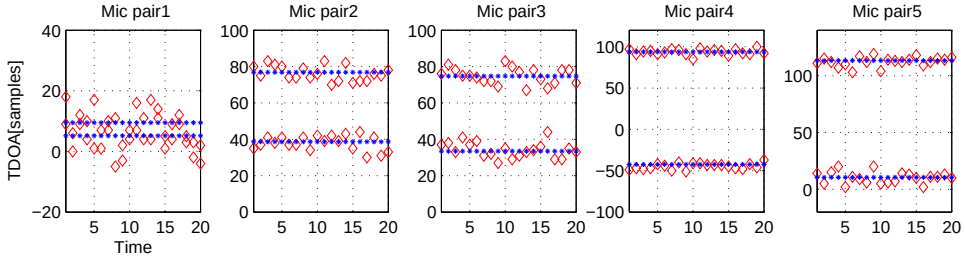
The first simulation is performed with additive Gaussian noise in the measurements. The variance of the noise for each sensor pair is $\sigma_z^2 = 10$ and $\sigma_z^2 = 20$. The measurements used in the simulation are shown in Figure 5.5. The blue asterisks show the true measurements, and the red diamonds show the measurements with errors.

Figure 5.6 shows the initialization results. The asterisks are the true positions of the sources, and the circles are the estimated positions of the sources. Even though the measurements are noisy, the detected initial positions are usually very close to the true positions.

The next simulation looks at the performance with missing data and falsely detected data. Figure 5.7 shows the measurements with falsely detected data and missing data. Falsely detected data do not affect the initialization much, as long as there are accurate measurements. Since the falsely detected data are generated independently of the true measurements, the data likelihood of any hypothesis including falsely detected data, is usually less than the data likelihood for a true association. Even with Gaussian noise with $\sigma_z^2 = 10$ and falsely detected data rate of $\lambda_f = 0.2$,



(a) Measurements with Gaussian noise with $\sigma_z^2 = 10$



(a) Measurements with Gaussian noise with $\sigma_z^2 = 20$

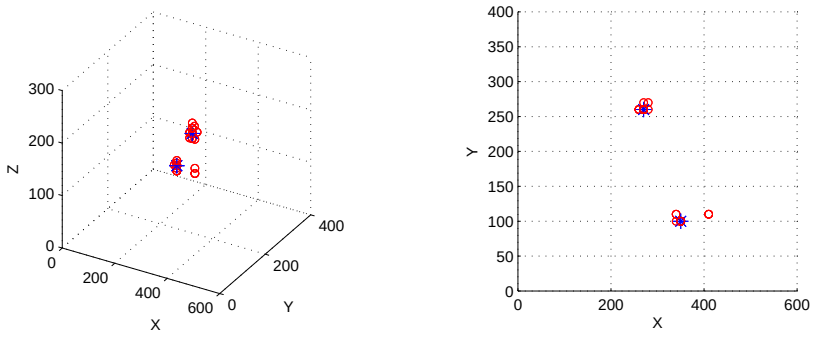
Figure 5.5: Measurements with different error conditions at two fixed targets. The blue asterisks are true measurements, and the red diamonds are the measurements with errors.

the percentage of correct initializations is greater than 95%. However, missing data affects the performance much more seriously.

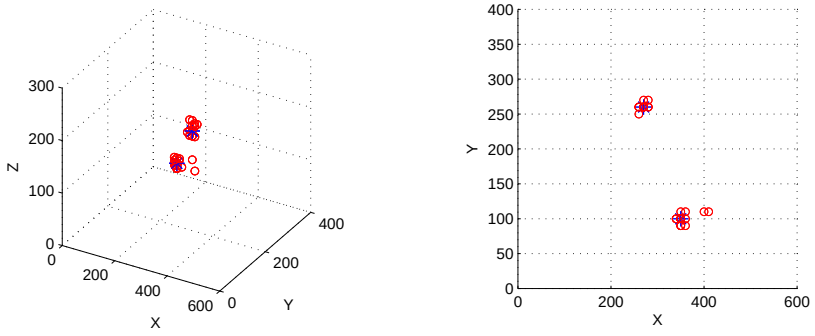
The performance of the initialization due to missing data degrades when there are multiple speakers. The main reason for this degradation is the estimation of the threshold, ζ_0 in Table 4.10. When one target has all true measurements and the other target loses several measurements, the data likelihood for the target that has all true measurements is much larger than the data likelihood of the target that has missing data. The ratio of their likelihoods can sometimes be greater than 10^3. Therefore, estimating the proper ζ_0 in different missing data environments is very difficult. When a small value for ζ_0 is used to include the second target, the final initialization has many incorrect positions. The simulation result using measurements with missing data at a rate of $\lambda_m = 0.2$ fails around 20% of the time.

Initialization for the Visual Tracking :

Additive Gaussian noise with variances of $\sigma_z^2 = 50$, $\sigma_z^2 = 100$, and $\sigma_z^2 = 400$ is added to the true measurements. These noise levels are reasonable because the center of a



(a)With measurements with Gaussian noise with $\sigma_z^2 = 10$



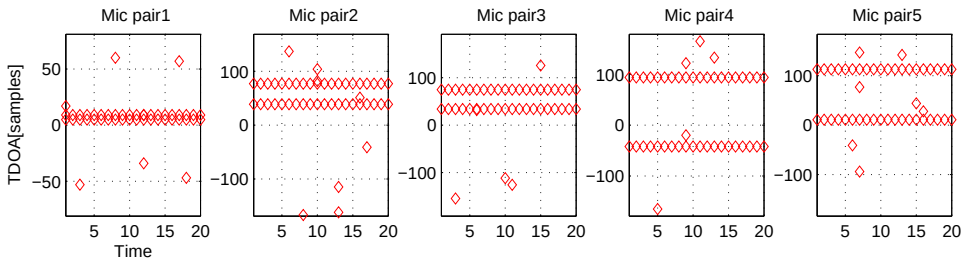
(b)With measurements with Gaussian noise with $\sigma_z^2 = 20$

Figure 5.6: Results of the initialization using different Gaussian measurement errors. The figures on the left show the results in the 3D space, and the figures on the right show the results in the $x - y$ plane.

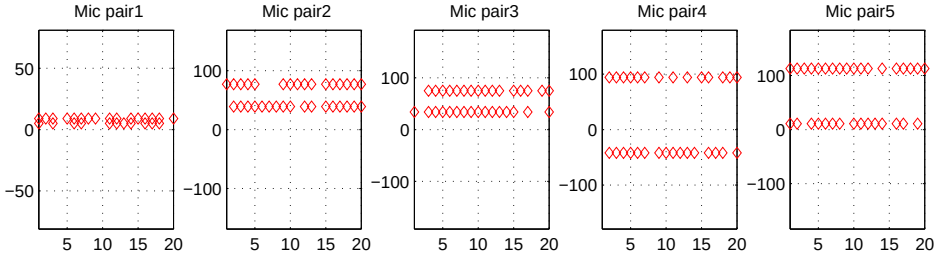
detected face always fluctuates within the face. For example, if the size of the face is 50 x 50, the maximum measurement error for this face can be 25 pixels, corresponding to $\sigma_z^2 \approx 64$. One snapshot of location measurements with Gaussian noise is shown in Figure 5.8. The blue asterisks indicate the true measurements, and the red dots represent the measurements with the Gaussian errors.

The initialization results are shown in Figure 5.9. The initialization is quite robust at finding the initial positions and the number of targets at a noise level of $\sigma_z^2 = 100$. However, the initialization sometimes fails at the noise level of $\sigma_z^2 = 400$. Gaussian noise with $\sigma^2 = 400$ is actually quite extreme; the natural noise level in our environment is close to $\sigma_z^2 = 100$.

The next simulation investigates the results of the initialization with missing data and falsely detected data. Initially, only missing or falsely detected data are applied to

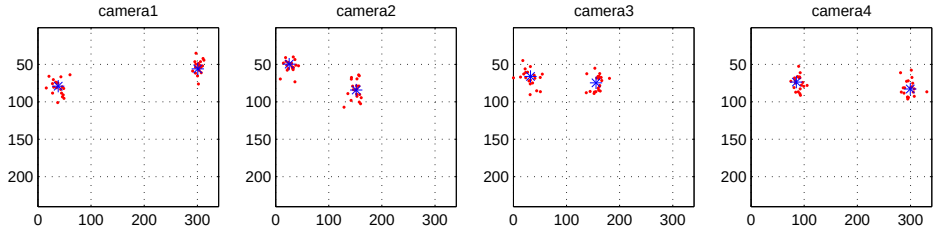


(a) Measurements with falsely detected data of $\lambda_f = 0.2$

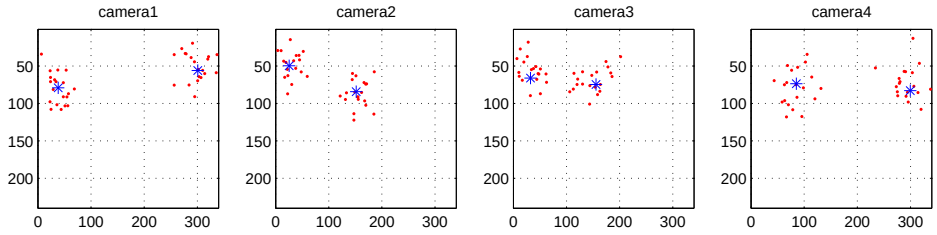


(b) Measurements with missing data of $\lambda_m = 0.2$

Figure 5.7: Measurements with falsely detected data and missing data without additive Gaussian noise used for the initialization.



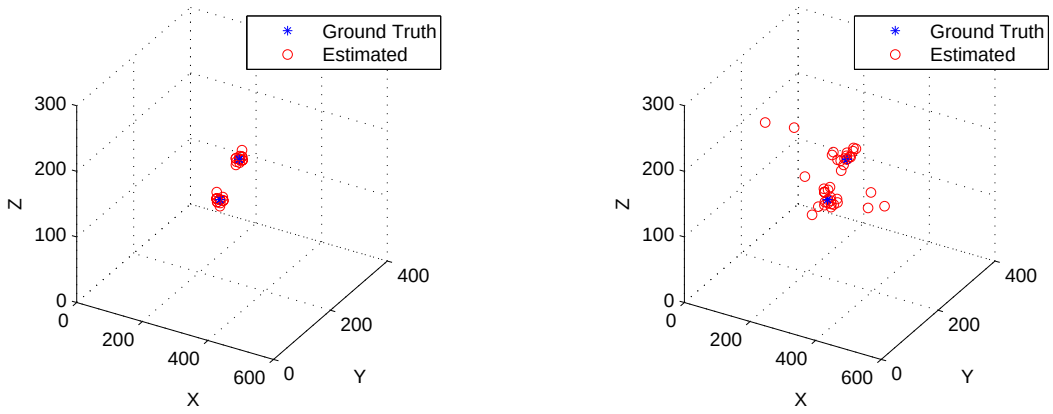
(a) $\sigma_z^2 = 100$



(b) $\sigma_z^2 = 400$

Figure 5.8: Measurements added with Gaussian noise with different variances.

the true measurements. As a result, the initialization for a target is done correctly if there are at least two reasonably accurate measurements from two cameras. It can be concluded, then, that initialization using visual measurements does not need multiple measurements from many cameras, but from only two cameras, which is fewer sensors

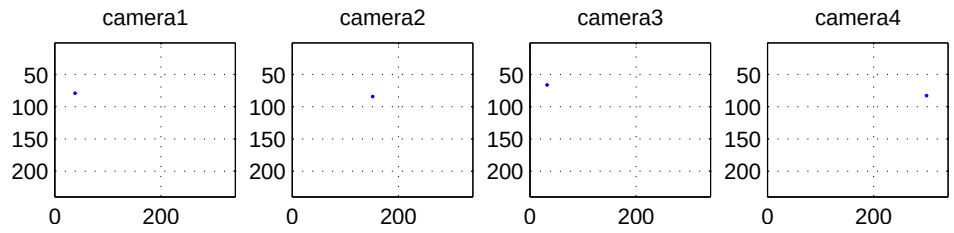


(a) $\sigma_z^2 = 100$ (b) $\sigma_z^2 = 400$

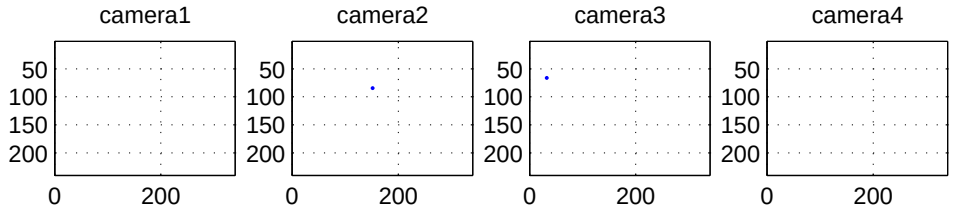
Figure 5.9: Initialization results with different Gaussian noise levels.

than speaker tracking requires. This reduction in the number of cameras is possible because a visual measurement from one camera contains three different pieces of information (two positions and a size) compared to only one piece of information from an acoustic measurement (a time delay difference). Even though this research uses the position as the visual measurement, the position still carries one more piece of information than an acoustic measurement.

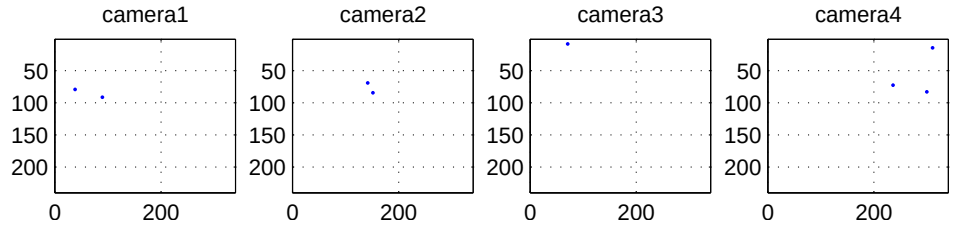
Falsely detected data do not affect the initialization much if each target has at least two accurate measurements. This is demonstrated in Figure 5.10 using target 1. Figure 5.10(a) shows correct measurements for target 1 from four cameras at one time instance. However, Figure 5.10(b) lacks two measurements from camera 1 and camera 4. Figure 5.10(c) has one missing measurement from camera 3 and four falsely detected data. For these three cases, the initialization successfully finds the initial position of the target, as in Figure 5.10(d). The difficult situation for the initialization is when measurement errors and missing data occur at the same time. When all measurements deviate from the true values, and when one or two measurements are missing, the initialization fails to find the initial position of the target.



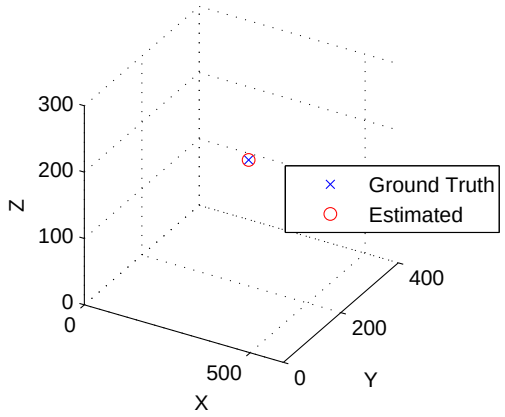
(a) Correct and complete measurements



(b) Measurements with two missing data



(c) measurements with one missing data and five falsely detected data



(d) Initialization results using all three measurements above

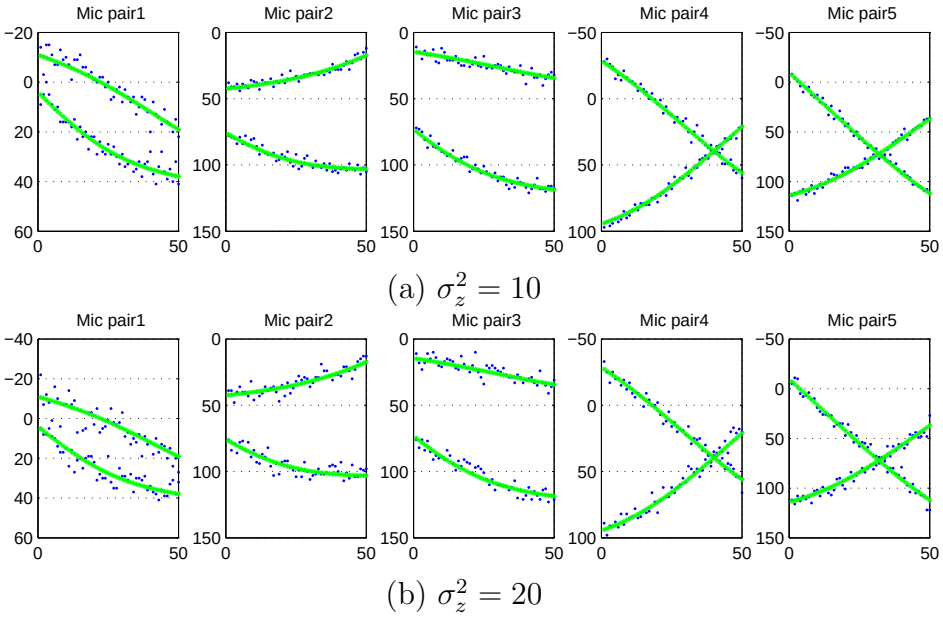
Figure 5.10: One result of the initialization using missing data and falsely detected data.

5.3.3 Tracking Multiple Targets

Tracking two speakers using microphones:

The performance for tracking two speakers is evaluated with Gaussian noise with $\sigma_z^2 = 10$ and $\sigma_z^2 = 20$. The measurements in these noise conditions are shown in 5.11.

Table 5.13 shows an MAE using DA-IPPF and MC-JPDAF for 20 Monte-Carlo repetitions. In the table, the results for a column for a single speaker are simulated by using only one target, either target 1 or target 2 indicated in the table. The MAE in parentheses in the table is in x, y, and z orders. The MAE values are in centimeters.



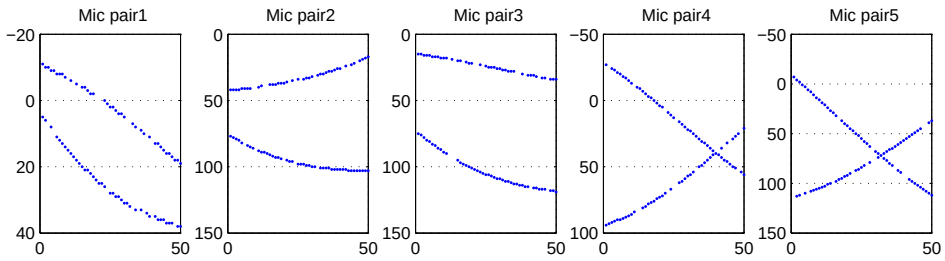
(a) $\sigma_z^2 = 10$

(b) $\sigma_z^2 = 20$

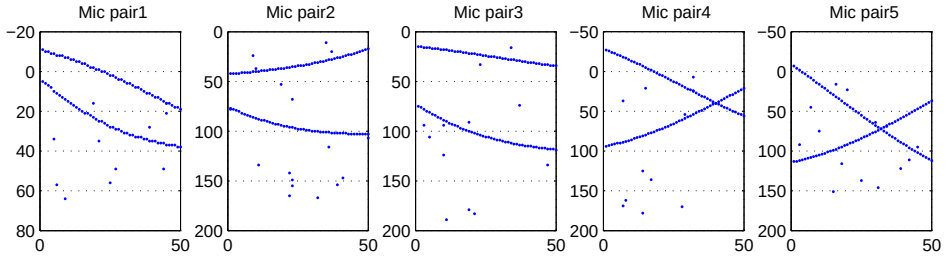
Figure 5.11: Sample measurements with additive Gaussian noise. The green lines show true measurements. The X axis is time, and the Y axis is TDOA values in samples.

Table 5.13: The MAE values for the tracking two speakers using DA-IPPF and MC-JPDAF.

Variance			single speaker (x,y,z)[cm]	two speakers (x,y,z)[cm]
$\sigma_z^2 = 0$	target1	DA-IPPF	(4.3,3.3,3.0)	(4.9,3.6,3.2)
		MC-JPDAF	(3.3,2.7,2.1)	(3.0,2.4,2.0)
	target2	DA-IPPF	(4.6,3.5,2.9)	(5.3,3.2,3.3)
		MC-JPDAF	(3.0,2.3,1.9)	(3.6,2.0,1.7)
$\sigma_z^2 = 10$	target1	DA-IPPF	(5.0,3.9,5.9)	(5.3,4.5,6.2)
		MC-JPDAF	(3.9,2.9,5.3)	(4.0,3.3,5.4)
	target2	DA-IPPF	(4.8,4.0,5.8)	(5.6,3.6,5.4)
		MC-JPDAF	(3.6,2.9,5.6)	(4.1,2.8,4.8)
$\sigma_z^2 = 20$	target1	DA-IPPF	(5.2,4.3,7.5)	(5.7,4.9,7.6)
		MC-JPDAF	(4.7,3.6,7.1)	(4.3,3.8,6.8)
	target2	DA-IPPF	(5.2,4.6,7.6)	(6.6,4.2,7.6)
		MC-JPDAF	(4.3,3.7,7.2)	(4.9,3.2,6.3)



(a) With missing data of $\lambda_m = 0.3$



(b) With false alarms of $\lambda_f = 0.3$

Figure 5.12: Sample measurements with missing data and falsely detected data.

Both DA-IPPF and MC-JPDAF using TDOA measurements can track two speakers. The performance of MC-JPDAF is slightly better than the performance of DA-IPPF. Since the performance degradation with an additive noise level with $\sigma_z^2 = 20$ is less than $1-2$ [cm] compared to the result with no noise, it can be concluded that the additive Gaussian noise with a variance less than $\sigma_z^2 = 20$ does not affect the performance significantly.

Under small additive Gaussian noise with $\sigma_z^2 = 20$ or less, the performance of tracking a single speaker is similar to that of two speakers. This indicates that tracking multiple targets does not degrade the overall performance unless there is a problem that causes in accurate data association.

Next, the performance of tracking two speakers is evaluated with falsely detected data and missing data. Figure 5.12 shows measurements with falsely detected data at an average rate $\lambda_f = 0.3$ and missing data at an average rate $\lambda_m = 0.3$. The missing data look like broken links in the figure. There is no available information about data association between targets and measurements.

Table 5.14 shows MAE values when additive Gaussian noise is combined with

falsely detected data. With the same Gaussian noise, even though the rate of falsely detected data is increased, the MAE values are almost the same. The observation is that falsely detected data do not significantly affect the performance of tracking for both data association methods.

Table 5.14: The MAE values for speaker tracking with falsely detected data.

			MC-JPDAF (x,y,z)[cm]	DA-IPPF (x,y,z)[cm]
$\lambda_f = 0.1$	$\sigma_z^2 = 0$	target1	(3.3,2.5,2.1)	(4.4,3.9,3.2)
		target2	(3.5,1.9,1.7)	(5.1,3.0,2.8)
	$\sigma_z^2 = 10$	target1	(4.2,3.4,5.5)	(5.4,4.5,6.2)
		target2	(4.3,2.8,4.8)	(5.4,3.6,5.3)
	$\sigma_z^2 = 20$	target1	(4.5,3.8,7.2)	(5.7,5.0,8.2)
		target2	(5.1,3.5,6.4)	(6.3,5.1,10.0)
$\lambda_f = 0.2$	$\sigma_z^2 = 0$	target1	(3.3,2.5,2.2)	(4.7,4.1,3.4)
		target2	(3.6,2.2,1.9)	(4.8,3.2,3.2)
	$\sigma_z^2 = 10$	target1	(4.0,3.3,5.3)	(5.1,4.6,6.0)
		target2	(4.4,2.7,4.9)	(5.8,4.0,6.0)
	$\sigma_z^2 = 20$	target1	(4.5,3.8,7.1)	(5.7,4.7,7.6)
		target2	(4.9,3.3,6.4)	(6.3,4.6,8.7)
$\lambda_f = 0.3$	$\sigma_z^2 = 0$	target1	(3.3,2.6,2.2)	(4.7,3.8,3.2)
		target2	(3.5,2.0,1.8)	(5.2,3.2,2.9)
	$\sigma_z^2 = 10$	target1	(4.0,3.3,5.5)	(5.1,4.3,6.3)
		target2	(4.4,2.8,4.9)	(5.9,4.3,6.9)
	$\sigma_z^2 = 20$	target1	(4.3,3.7,6.9)	(5.6,4.9,7.5)
		target2	(5.1,3.4,6.2)	(6.5,4.6,7.9)

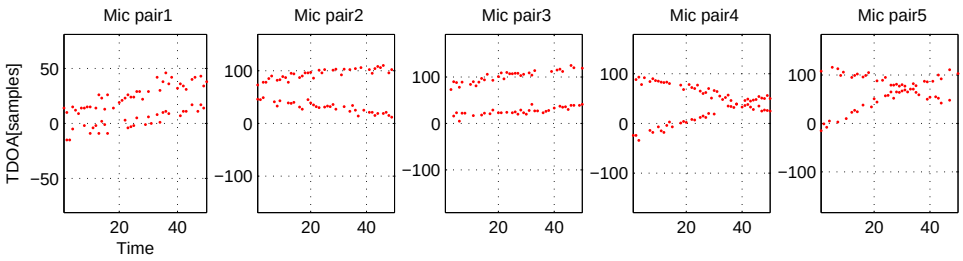
Table 5.15 shows MAE with missing data and different additive measurement noise. While the MAE values of MC-JPDAF do not change much according to the increase of the missing data rate, DA-IPPF has very high MAE values at $\lambda_m = 0.2$ and $\sigma_z^2 = 10$ for target 2 and $\lambda_m = 0.3$ and $\sigma_z^2 = 10$ for both targets. This performance decrease in DA-IPPF occurs when the measurements of multiple targets are very close. Since this condition was anticipated, the association parameter for each target in DA-IPPF was proposed to be assigned different values. However, resampling for each target creates the same problem. Since resampling is independently

Table 5.15: The MAE values for tracking two speakers with missing data.

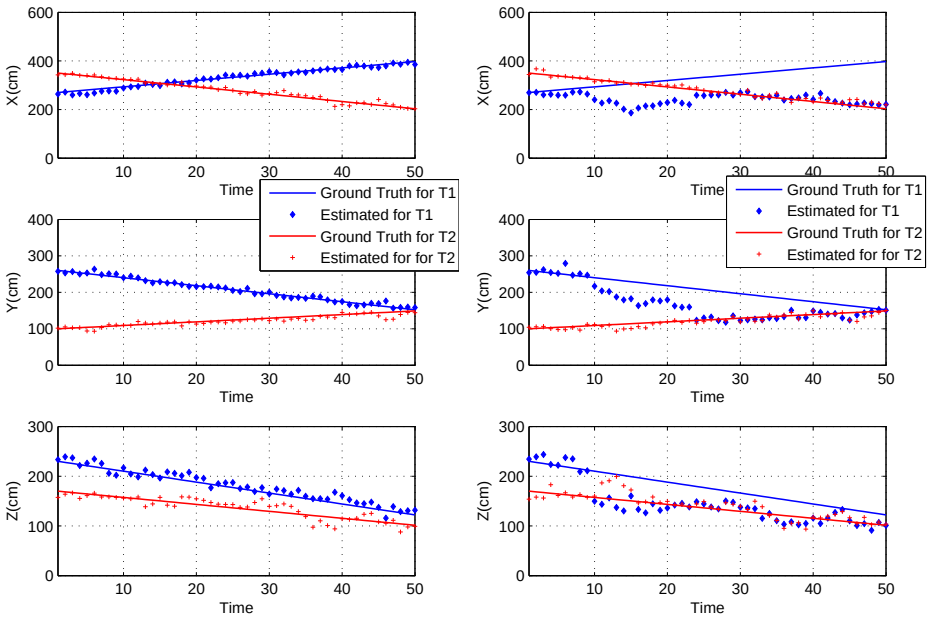
			MC-JPDAF (x,y,z)[cm]	DA-IPPF (x,y,z)[cm]
$\lambda_m = 0.1$	$\sigma_z^2 = 0$	target1	(3.5,2.9,2,3)	(5.1,3.9,3.4)
		target2	(3.8,2.2,1.9)	(5.9,4.1,5.8)
	$\sigma_z^2 = 10$	target1	(4.0,3.1,5.4)	(5.4,4.8,5.9)
		target2	(4.4,2.8,4.8)	(5.9,4.6,7.9)
$\lambda_m = 0.2$	$\sigma_z^2 = 0$	target1	(4.1,3.5,2.3)	(5.6,4.8,3.7)
		target2	(4.6,2.5,1.9)	(5.9,4.7,7.4)
	$\sigma_z^2 = 10$	target1	(4.1,3.5,5.4)	(5.6,5.1,6.3)
		target2	(4.7,2.9,4.8)	(11.6,6.8,12.4)
$\lambda_m = 0.3$	$\sigma_z^2 = 0$	target1	(3.9,3.2,2.5)	(5.4,4.8,4.0)
		target2	(4.7,2.6,2.2)	(6.8,5.1,7.7)
	$\sigma_z^2 = 10$	target1	(4.5,3.7,5.6)	(11.1,7.4,8.3)
		target2	(5.1,3.0,4.9)	(12.5,6.9,13.9)

performed based on the probability of $r_{k,t}^{(n)}$, the association parameters for both targets become the same after the resampling at each target. An example of this is shown in Figure 5.13. In a noisy environment with additive Gaussian noise with $\sigma_z^2 = 20$ and missing data of $\lambda_m = 0.3$, the measurements around t=1 $\sim$ 5 from Mic pair 1 and Mic pair 2 are very close. This simulation condition does not always generate this problem: it occurred only a few times among many Monte-Carlo simulation repetitions. With these measurements, association parameters for target 1 become the values of the association parameters for target 2 after resampling. Even though the measurements are separated, the states for both targets track the same target. This indicates that the data association for multiple targets cannot be accomplished without considering the dependence between targets. As a solution, the resampling process in each target can be deleted. However, DA-IPPF without resampling incurs the curse of dimensionality, which results in a general decrease of the entire tracking performance with the same number of particles. However, the problem in DA-IPPF does not occur with MC-JPDAF.

Tracking two people using multiple cameras:



(a) Measurements with missing data of $\lambda_m = 0.3$ and $\sigma_z^2 = 20$



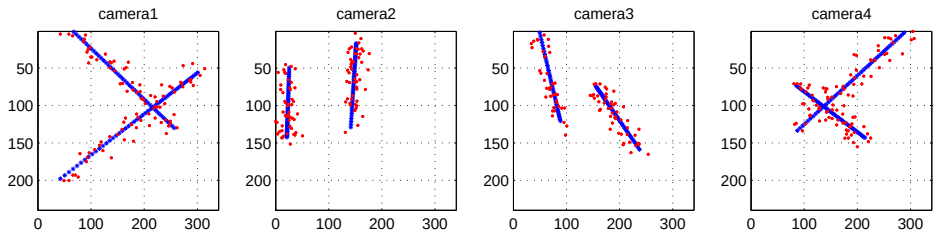
(b) The result using MC-JPDAF (c) The result using DA-IPPF

Figure 5.13: One result using DA-IPPF.

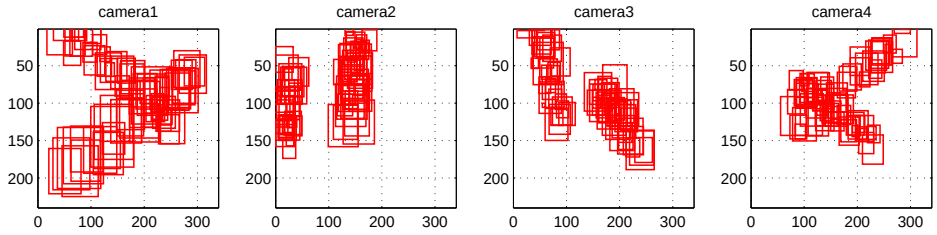
Using the same tracking scenario in Figure 5.4, the visual measurements, locations and a size, with Gaussian noise with $\sigma_z^2 = 100$, are shown in Figure 5.14.

The performance according to the number of cameras is evaluated. Table 5.16 shows the MAE of single-target tracking using target 1 with no noise and different white Gaussian noise levels with $\sigma_{\mathbf{z}}^2 = 10$ and $\sigma_{\mathbf{z}}^2 = 100$ using from one to four cameras.

As shown in Table 5.16, tracking a point in 3D using one camera is not accurate even if location and size measurements are used. Therefore, using more than one camera is recommended to track targets in 3D. However, surprisingly, the performance using three cameras or even four cameras does not offer much improvement over using



(a)Measurements of positions, $\mathbf{z_L}$



(b)Measurements of sizes, $\mathbf{z_w}$

Figure 5.14: measurements for two people with additive Gaussian noise with $\sigma_z^2 = 100$ and $\lambda_v = 0.2$. The solid, blue lines show the true positions. The red dots and rectangles show the measurements with errors.

Table 5.16: The MAE values for tracking target 1 using different numbers of cameras.

Variance	One camera (x,y,z)[cm]	Two cameras (x,y,z)[cm]	Three cameras (x,y,z)[cm]	Four cameras (x,y,z)[cm]
$\sigma_z^2 = 0$	(27.2,33.6,5.8)	(2.4,2.3,1.6)	(2.4,1.8,1.4)	(2.3,1.5,1.3)
$\sigma_z^2 = 10$	(26.1,24.2,4.5)	(3.1,2.8,2.1)	(3.1,2.3,1.9)	(2.7,1.9,1.7)
$\sigma_z^2 = 100$	(31.4,28.4,7.8)	(7.7,7.4,5.0)	(7.0,5.3,4.4)	(6.4,4.8,4.0)

two. Thus, it can be concluded that two cameras are sufficient to track 3D locations of people. From this result, the advantage of using more than two cameras occurs only when the environment has a high missing data rate. An example of the case when missing data might be extremely high is when a target is out of the view of some cameras. In this case, if any two cameras can capture this target and detect a visual feature for the target, the target can be tracked.

Next, the performance using different data association methods is evaluated with different values of additive Gaussian noise. Tables 5.17 and 5.18 show the MAE values. The MAE of the MC-JPDAF is slightly better than the MAE of the DA-IPPF in the case of no measurement noise, but the MAE of the DA-IPPF is slightly better than the MAE of the MC-JPDAF when measurement noise is present. Here, even

though measurements for both targets from camera 1 and 4 appear to be crossing in Figure 5.14, in fact they do not cross. There are time differences between the measurements. They are drawn in the x-y plane without identifying the time axis.

Table 5.17: The MAE values for tracking two people using MC-JPDAF.

Variance		one target (x,y,z)[cm]	two targets (x,y,z)[cm]
$\sigma_z^2 = 0$	target1	(2.4,1.7,1.4)	(2.2,1.6,1.4)
	target2	(2.2,1.8,1.3)	(2.4,1.6,1.3)
$\sigma_z^2 = 50$	target1	(4.6,3.2,2.9)	(4.4,3.3,2.7)
	target2	(4.6,3.2,2.7)	(4.7,3.4,2.7)
$\sigma_z^2 = 100$	target1	(6.8,4.7,3.7)	(7.4,4.9,4.0)
	target2	(6.2,4.6,4.2)	(6.8,4.9,4.2)

Table 5.18: The MAE values for tracking two people using DA-IPPF.

Variance		one target (x,y,z)[cm]	two targets (x,y,z)[cm]
$\sigma^2 = 0$	target1	(3.3,2.3,2.0)	(3.4,2.5,2.1)
	target2	(3.2,2.2,2.0)	(3.3,2.3,1.9)
$\sigma_z^2 = 50$	target1	(4.3,3.1,2.8)	(4.6,3.3,2.9)
	target2	(4.2,3.1,2.7)	(4.1,3.0,2.6)
$\sigma_z^2 = 100$	target1	(5.0,3.9,3.4)	(5.2,4.1,3.5)
	target2	(5.1,3.9,3.4)	(5.2,3.7,3.3)

The performance using missing data and falsely detected data both with and without additive measurement noise is evaluated in Tables 5.19 and 5.20. Here, DA-IPPF is better than MC-JPDAF most of the time. This is because there are three visual measurements: two positions and a size. The size measurement helps to associate accurate/appropriate association parameters when the position measurements are similar.

Table 5.19: The MAE values for tracking two people with missing data.

			MC-JPDAF (x,y,z)[cm]	DA-IPPF (x,y,z)[cm]
$\lambda_m = 0.1$	$\sigma_z^2 = 0$	target1	(2.5,1.7,1.4)	(3.7,2.6,2.3)
		target2	(2.6,1.7,1.4)	(3.6,2.5,2.1)
	$\sigma_z^2 = 50$	target1	(4.4,3.5,2.9)	(4.9,3.9,3.1)
		target2	(4.7,3.3,2.8)	(4.7,3.4,2.7)
	$\sigma_z^2 = 100$	target1	(7.3,5.3,4.5)	(5.6,3.9,3.6)
		target2	(6.9,4.9,3.8)	(5.3,3.8,3.4)
$\lambda_m = 0.2$	$\sigma_z^2 = 0$	target1	(2.5,1.7,1.5)	(4.1,2.7,2.4)
		target2	(2.7,1.7,1.3)	(4.2,2.8,2.5)
	$\sigma_z^2 = 50$	target1	(5.1,3.8,3.2)	(4.8,3.6,3.0)
		target2	(4.8,3.6,2.7)	(4.9,3.4,3.1)
	$\sigma_z^2 = 100$	target1	(7.5,5.7,4.7)	(5.4,4.3,3.8)
		target2	(6.9,5.0,3.9)	(5.3,4.1,3.7)
$\lambda_m = 0.3$	$\sigma_z^2 = 0$	target1	(2.6,1.7,1.5)	(4.3,2.9,2.6)
		target2	(2.7,1.8,1.4)	(4.0,2.7,2.7)
	$\sigma_z^2 = 50$	target1	(5.5,4.0,3.5)	(4.9,3.9,3.3)
		target2	(5.0,3.8,2.9)	(4.9,3.6,3.4)
	$\sigma_z^2 = 100$	target1	(7.6,5.6,4.8)	(6.0,4.5,3.9)
		target2	(7.2,5.3,4.1)	(5.6,4.2,3.8)

5.3.4 Tracking a Single Speaker among Multiple People using Multiple Microphones

This section simulates tracking a single speaker when the current speaker switches among multiple people. The first scenario is when the current speaker switches between two people who are moving constantly, as in Figure 5.4. Since the current speaker switches at time $t = 31$ from target 1 to target 2, as in Figure 5.15(b), the resulting trajectory of the speaker is as shown in Figure 5.15(a). The next simulation is for a scenario when the current speaker alternates among three people at a fixed position more frequently than in the first scenario, which is a more likely scenario in a conference environment.

The problem in speaker switching is that the predefined state update model is no longer valid. Therefore, the proposal function using the state update model with a

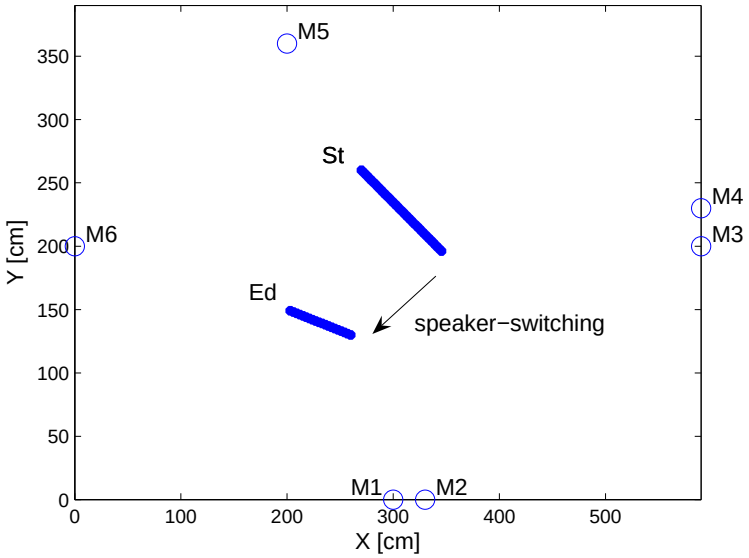
Table 5.20: The MAE values for tracking two people with falsely detected data.

			MC-JPDAF (x,y,z)[cm]	DA-IPPF (x,y,z)[cm]
$\lambda_f = 0.1$	$\sigma_z^2 = 0$	target1	(2.3,1.5,1.4)	(3.3,2.4,1.9)
		target2	(2.4,1.6,1.3)	(3.3,2.3,1.9)
	$\sigma_z^2 = 10$	target1	(4.9,3.6,3.1)	(4.4,3.2,2.8)
		target2	(4.5,3.5,2.6)	(4.3,3.1,2.5)
	$\sigma_z^2 = 20$	target1	(7.1,4.9,4.2)	(5.3,3.8,3.4)
		target2	(6.6,4.7,4.1)	(4.7,3.8,3.3)
$\lambda_f = 0.2$	$\sigma_z^2 = 0$	target1	(2.2,1.6,1.3)	(3.4,2.4,2.2)
		target2	(2.4,1.6,1.3)	(3.4,2.3,1.9)
	$\sigma_z^2 = 10$	target1	(4.8,3.7,3.0)	(4.3,3.4,2.8)
		target2	(4.8,3.5,2.6)	(4.2,3.3,2.5)
	$\sigma_z = 20$	target1	(6.7,5.2,4.4)	(4.9,3.9,3.5)
		target2	(6.4,4.6,3.6)	(4.9,3.7,3.1)
$\lambda_f = 0.3$	$\sigma_z^2 = 0$	target1	(2.3,1.7,1.4)	(3.4,2.3,2.0)
		target2	(2.4,1.6,1.3)	(3.3,2.4,1.9)
	$\sigma_z^2 = 10$	target1	(4.6,3.5,2.8)	(4.3,3.2,2.9)
		target2	(4.6,3.5,2.8)	(4.3,3.1,2.7)
	$\sigma_z^2 = 20$	target1	(6.5,5.3,4.6)	(5.2,3.9,3.5)
		target2	(7.4,5.0,3.9)	(5.2,3.9,3.4)

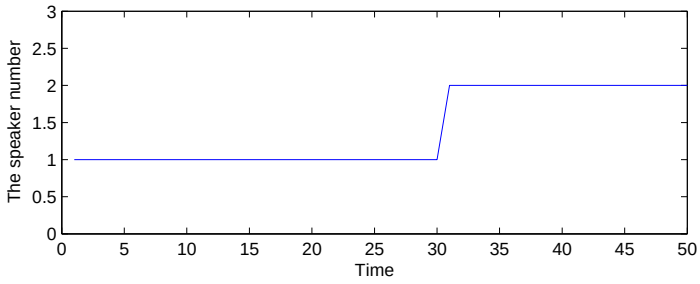
small variance cannot distribute particles to the location of a new target. As a solution, a different proposal function, like a Gaussian mixture model of Equation (66), should be used. This research uses the following Gaussian mixture model:

$$q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t) \sim \frac{1}{N_t} p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}) + \sum_{i=1}^{5} \sum_{j=1}^{M^i} \frac{1}{N_t} p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}) p(\mathbf{z}_{j,t}^i|\mathbf{x}_{k,t}).$$

Here, the same weight is applied to every mixture, so N_t is the total number of measurements at time=t plus 1 for the state update model as $1 + \sum_{i=1}^{5} M^i = N_t$. However, even when this Gaussian mixture is used for the proposal function, $p_k^i(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}) p(\mathbf{z}_{j,t}^i|\mathbf{x}_{k,t})$ cannot push the new particles into the position of a new speaker because $\mathbf{x}_{k,t-1}^{(n)}$ is not distributed throughout the whole support region but only in the most likely location for the old speaker. Therefore, $p_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)})$ in the posterior probability should have a large variance to stretch the particles over the



(a) The trajectory of an actual speaker



(b) The speaker number according to the time

Figure 5.15: The ground truths of a current speaker. The actual speaker is switched from target 1 to 2 at $t = 31$.

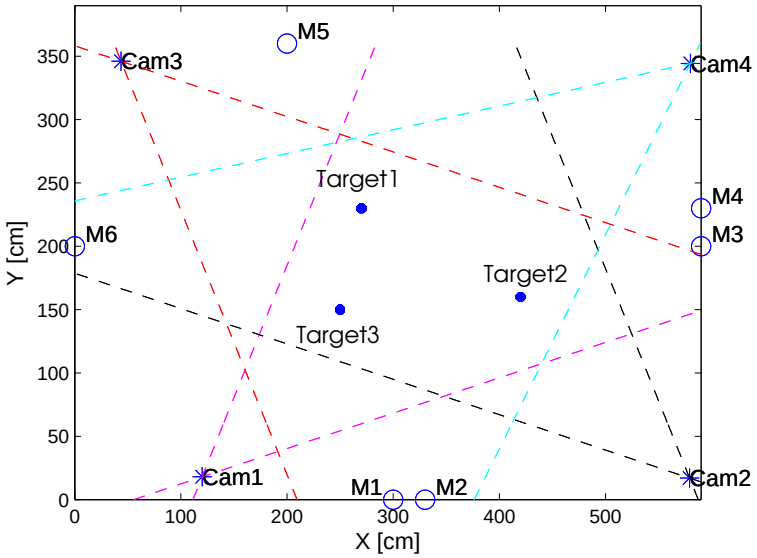
entire support region as

$$q_k(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}^{(n)}, \mathbf{z}_t) = \frac{1}{N_t} N(\mathbf{x}_{k,t}; \mathbf{x}_{k,t-1}^{(n)}, \Sigma_1) + \sum_{i=1}^{5} \sum_{j=1}^{M^i} \frac{1}{N_t} N(\mathbf{x}_{k,t}; \mathbf{x}_{k,t-1}^{(n)}, \Sigma_2) p(\mathbf{z}_{j,t}^i|\mathbf{x}_{k,t})$$

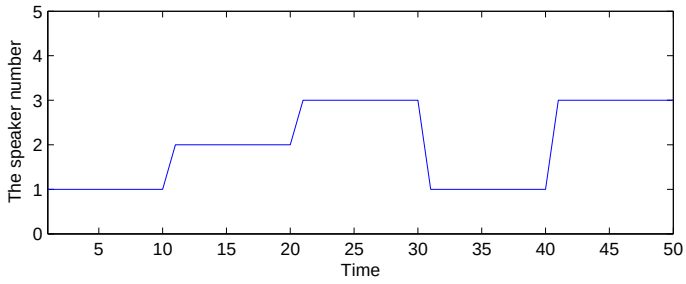
with

$$\Sigma_1 = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 50^2 & 0 & 0 \\ 0 & 50^2 & 0 \\ 0 & 0 & 30^2 \end{bmatrix}$$

The result applied with this proposal function using MC-JPDAF is shown in Figure 5.17(a) and (b). These two results have performance degradation at the time



(a) The positions of three people in the fixed positions



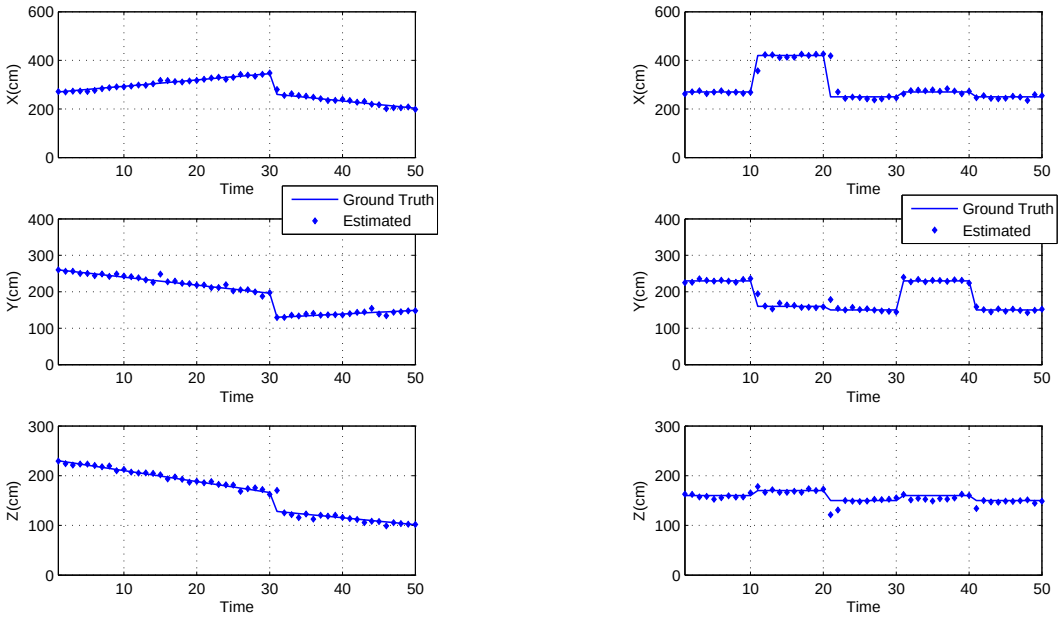
(b) The speaker number according to the time

Figure 5.16: The ground truths of a current speaker. The actual speaker is changed in the orders of target 1, 2, 3, 1, and 3.

the actual speaker changes, but the tracking follows the new speaker very quickly after one or two time slots.

5.3.5 Discussion

This chapter proposed a state space, a state update model, and a data likelihood function for both multiple-speaker tracking using only microphones and multiple people tracking using multiple cameras. The initialization algorithm for two targets was tested with several different error conditions such as additive Gaussian noise, missing data, and falsely detected data. For the tracking simulation, the two data association methods in Chapter 4 were applied when the measurements had error conditions



(a) A speaker switch from target 1 to target 2 (b) Speaker switches among three people

Figure 5.17: The results of speaker switching when there is no measurement error.

similar to those in the initialization stage.

For acoustic tracking, MC-JPDAF was more robust than DA-IPPF over all different noise conditions, especially when there was missing data. DA-IPPF was found to have serious performance degradation when measurements cross. This indicates that the data association for multiple targets cannot be accomplished without considering the dependence between targets. As a solution, DA-IPPF does not use resampling for each target, but DA-IPPF without the resampling decreases the overall performance in general.

Speaker switching among multiple people was also tested. The tracking for speaker switching among multiple people using microphones is a critical matter because the state-state model is no longer valid. A Gaussian mixture model with different variances was applied as the proposal function. Tracking using the Gaussian mixture worked quite well when there was speaker switching even though there is a performance degradation when the actual speaker was changed.

When tracking multiple people using multiple cameras, under ideal conditions with a high detection rate, an increase in the number of cameras above two does not significantly improve performance. However, with a high missing data rate, the presence of additional cameras (three or more) improves the robustness of the entire performance because it incases the likelihood of two sets of camera measurements at any one time.

DA-IPPF for visual tracking shows slightly better performance than MC-JPDAF. This is because visual tracking uses more visual features than the acoustic features in the acoustic tracking. When there are more measurements, the association of the measurements to the true, hidden targets is easier. This demonstrates that the data association problem in DA-IPPF when the measurements are very close can be helped by using different, multiple features. However, DA-IPPF still has the problem of assigning the same measurement to multiple targets, which occurs in speaker tracking when there is a high missing data rate and additive Gaussian noise when measurements cross.

Missing data generally degrades the performance more in both the initialization and tracking than do false alarms. Therefore, if both cannot be avoided at the same time, increasing the number of falsely detected data is preferable to increasing the amount of missing data.

CHAPTER VI

PARTICLE FILTERING FOR MULTIPLE PEOPLE TRACKING AND SPEAKING ACTIVITY DETECTION USING MULTIPLE CAMERAS AND MICROPHONES

6.1 Overview

As introduced in Section 2.6, sensor fusion aims to use redundancy and complementarity from different sensors to obtain more robust performance in a non-ideal environment. For instance, in the application of audio-visual tracking, visual features contribute to improve the performance under a severely noised or reverberant environment for speaker tracking, and acoustic features can give more clues for the position of the current speaker in cases of occlusion.

The block diagram of the joint audio-visual people tracking system is given in Figure 6.1.

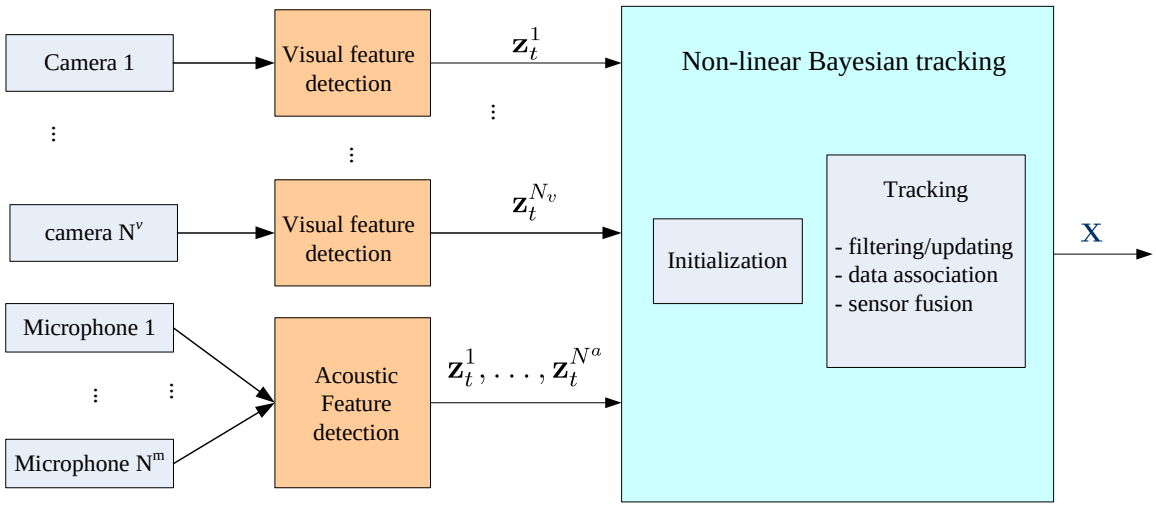


Figure 6.1: The block diagram of joint audio-visual tracking.

Since there is no guarantee that acoustic data exists in the initialization stage, the initialization for the joint tracking is performed using visual data alone, which is the same as the initialization for the visual tracking in Chapter 5.

6.2 Data Models

The state space is defined as the locations of multiple people, which is the same as for the speaker tracking and the visual tracking system, but augmented with a speaker activity flag to indicate whether or not each person is speaking. The whole state space has two components, a location and a speaking activity:

$$[\chi_{1,t} \cdots \chi_{k,t} \cdots \chi_{K,t}] = [\mathbf{x}_t, \mathbf{s}_t] \tag{89}$$

where $\chi_{k,t}$ is a state space for person k and is defined as $\chi_{k,t} = [\mathbf{x}_{k,t} s_{k,t}]$. $\mathbf{x}_{k,t}$ is the position in Cartesian coordinates, and $s_{k,t}$ is a one-bit flag, which is 1 if person k is detected as a speaker and 0 otherwise.

The state dynamics for $\mathbf{x}_{k,t}$ is a random walk with Gaussian noise as in Equation (78), but even though $s_{k,t}$ belongs to the state space, it does not evolve in time using a state update model as the other sub-states do, but is instead estimated.

The data likelihood is the product of each data likelihood in Equations (80) and (81) using independence assumption between the two different types of sensors, i.e., cameras and microphones:

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{x}_t) =& p(\mathbf{z}_{a,t}, \mathbf{z}_{v,t}|\mathbf{x}_t) \\ =& p(\mathbf{z}_{a,t}|\mathbf{x}_t) p(\mathbf{z}_{v,t}|\mathbf{x}_t) \qquad (90) \\ =& \prod_{i=1}^{N^a} p(\mathbf{z}_{a,t}^i|\mathbf{x}_t) \prod_{j=1}^{N^v} p(\mathbf{z}_{v,t}^j|\mathbf{x}_t) \qquad (91) \end{aligned}$$

The relative contribution of each sensor type can be changed using

$$p(\mathbf{z}_t|\mathbf{x}_t) = p(\mathbf{z}_{a,t}|\mathbf{x}_t)^{\alpha} p(\mathbf{z}_{v,t}|\mathbf{x}_t)^{1-\alpha}. \tag{92}$$

α is determined empirically and is less than 1.

6.3 Detection of Speaker Activity

A least square error is used to detect a single speaker as

$$s = \arg\min_{k} \left\{ \sum_{i=1}^{N^a} \sum_{n=1}^{N} (\mathbf{z}_{a,t}^{i} - \hat{\mathbf{z}}_{a}^{i}(\mathbf{x}_{k,t}^{(n)}))^2 \right\}, k = 1, ..., K.$$

This is the same as using the maximum data likelihood when the data likelihood is a Gaussian distribution. Unfortunately, this is possible only when the data association is known, which is not the assumption in this research. In the framework of data association, the detection of the speaking activity for each target is carried out without any extra estimation because the data association inherently includes this information in its framework, which is a target-to-measurement association parameter, $r_k^{i(n)}$.

In MC-JPDAF, $r_k^{i(n)}$ cannot be seen directly because it is wrapped in the posterior probability of target k to measurement j from sensor i, β_{jk}^{i} in Equation (55). If β_{0k}^{i} is smaller than $\{\beta_{jk}^{i}, j \neq 0\}$, then target k is determined to be a speaker.

$$s_{k,t} = \begin{cases} 0 & \text{if } \sum_{i=1}^{N^a} \beta_{0k}^{i} > \sum_{i=1}^{N^o} \{\beta_{jk}^{i}, j \neq 0\} \\ 1 & \text{otherwise} \end{cases}$$

In DA-IPPF, $r_k^{i(n)}$ is generated for each target, each sensor, and each particle and then is resampled, as in Table 4.8. After resampling for each target, all trivial, negligible $r_k^{i(n)}$s are deleted. The distribution of $\{r_k^{i(n)}\}$ for the acoustic sensors after resampling then shows the speaking activity of target k.

$$s_{k,t} = \begin{cases} 0 & \text{if } \sum_{i=1}^{N^a} \sum_{n=1}^{N} (r_k^{i(n)} == 0) > \eta \\ 1 & \text{otherwise} \end{cases}$$

6.4 Simulation Results

6.4.1 Simulation Environment

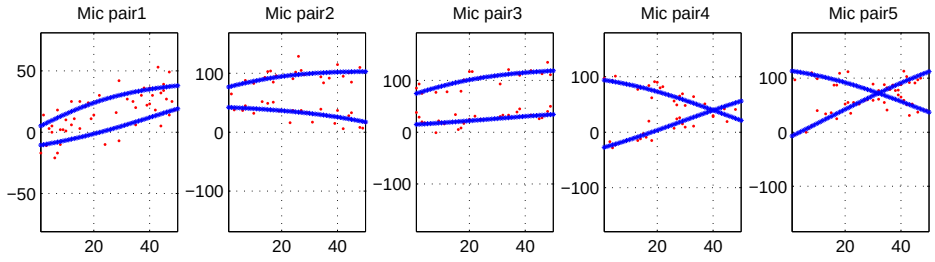
The simulation for joint tracking is performed using the same scenario in Figure 5.4 that was used for speaker tracking and visual tracking, but this simulation uses much

higher missing data rates and increased additive Gaussian noise ($\lambda_{m,a} = 0.5$, $\sigma_a^2 = 50$, $\lambda_{m,v} = 0.5$, $\sigma_v^2 = 400$). $\lambda_{m,a}$ is the parameter for the Poisson process governing the missing data for acoustic measurements, and $\lambda_{m,v}$ is the corresponding parameter for visual measurements. The missing data rate with $\lambda_m = 0.5$ indicates that half of the true measurements are lost. Since MC-JPDAF showed more stable results than DA-IPPF, the simulation here uses only MC-JPDAF.

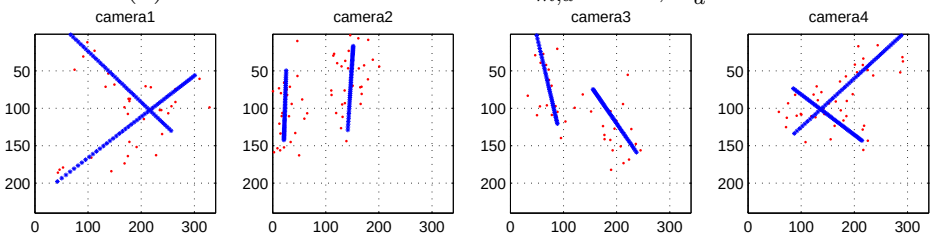
6.4.2 Joint Tracking by using Audio-visual Measurements

Sample measurements with this worst condition are shown in Figure 6.2. The solid blue lines show the true measurements, and the red dots show the actual measurements with Gaussian noise and missing data.

In Table 6.21, the result of the joint tracking is compared with that of speaker tracking and visual tracking using MC-JPDAF.



(a) Acoustic measurements: $\lambda_{m,a} = 0.5$, $\sigma_a^2 = 100$



(a) Visual measurements $\lambda_{m,v} = 0.5$, $\sigma_v^2 = 400$

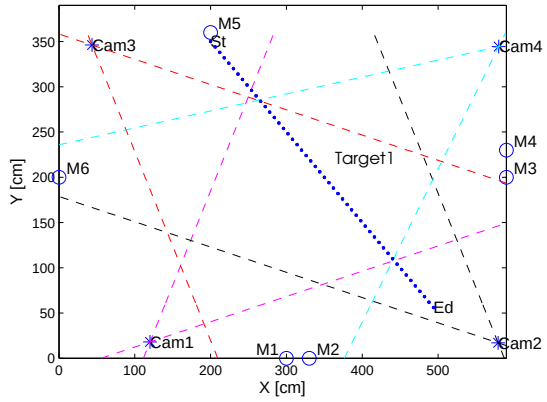
Figure 6.2: Sample measurements with an increased missing data rate and additive Gaussian noise. The solid lines are the true measurements, and the dots are measurements with errors and missing data.

Table 6.21: The performance of joint tracking with a high missing data rate and additive Gaussian noise.

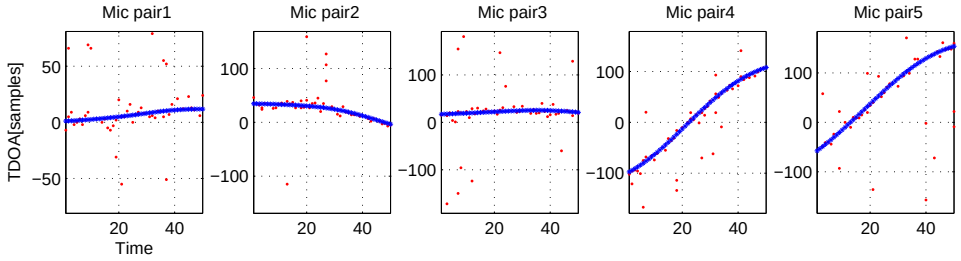
Error condition		Acoustic tracking (x,y,z)[cm]	Visual tracking (x,y,z)[cm]	Joint tracking (x,y,z)[cm]
$\lambda_{m,a} = 0.2$, $\sigma_a^2 = 20$	target 1	**(5.5,4.6,7.5)**	(24.3,19.5,13.5)	**(5.8,5.1,7.2)**
$\lambda_{m,v} = 0.5$, $\sigma_v^2 = 400$	target 2	**(5.6,3.9,6.5)**	(37.7,20.2,13.2)	**(6.3,4.3,6.3)**
$\lambda_{m,a} = 0.5$, $\sigma_a^2 = 100$	target 1	(13.8,11.1,15.6)	**(7.9,5.7,5.0)**	**(7.0,5.4,5.5)**
$\lambda_{m,v} = 0.2$, $\sigma_v^2 = 100$	target 2	(12.8,9.2,13.8)	**(7.5,5.2,4.4)**	**(6.5,4.8,5.1)**
$\lambda_{m,a} = 0.5$, $\sigma_a^2 = 100$	target 1	(13.8,11.1,15.6)	(24.3,19.5,13.5)	**(10.6,8.5,11.6)**
$\lambda_{m,v} = 0.5$, $\sigma_v^2 = 400$	target 2	(12.8,9.2,13.8)	(37.7,20.2,13.2)	**(10.7,7.3,10.3)**

In Table 6.21, the performance of joint tracking is still robust if only one measurement modality is reasonably good. In the first row of the table, the visual measurements have missing data with $\lambda_{m,v} = 0.5$ and additive Gaussian noise with $\sigma_v^2 = 400$, but the acoustic measurements are more reliable than the visual measurements. The MAE of joint tracking is much better than that for visual tracking and almost the same as that for acoustic tracking. The results in the second row, with opposite reliability, have almost the same result. When both measurements are severely contaminated, as in the third row of the table, the result of the joint tracking is better than that of either result using only one kind of sensor.

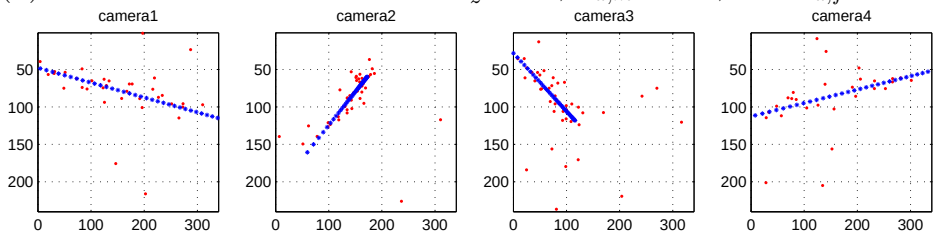
The next simulation scenario, which is shown in Figure 6.3, also shows the advantage of joint tracking compared to visual tracking alone. The trajectory of one person intentionally goes out of the view of multiple cameras. The beginning of the person's trajectory is shown only in camera 2. Therefore, visual tracking does not work well. However, joint tracking using acoustic and visual signals works quite well. The MAE using 20 Monte-Carlo repetitions using the condition of Figure 6.3 is $(10.6, 9.7, 11.1)$. This is worse than the results in Table 6.21, but the person can be tracked by using joint acoustic-visual measurements.



(a) Trajectory of a single person in the x-y plane



(b) Acoustic measurements with $\sigma_z^2 = 50$, $\lambda_{a,m} = 0.2$, and $\lambda_{a,f} = 0.2$



(c) Visual measurements with $\sigma_z^2 = 100$, $\lambda_{v,m} = 0.1$, and $\lambda_{v,f} = 0.2$

Figure 6.3: An example of an object missing/lost from cameras. The object is not seen at the beginning and ending of the trajectory in some cameras. The blue dots indicate true measurements, and the red dots indicate actual measurements.

6.4.3 Tracking of One Speaker among Multiple People

The next simulation is performed for sensor fusion in cases in which the measurement belongs only to a sub-state, not to the whole state. Only one person is assumed to talk among two people, either target 1 or target 2. The results are shown in Table 6.22. The speaker is marked with an asterisk in the table. The target, which has joint measurements (marked with an asterisk), is tracked much better than the other target. Even though the acoustic measurements have errors, the target, which

has joint measurements (shown in the third and fourth rows) tracks better. Therefore, this demonstrates that in a non-ideal environment, tracking performance using joint measurements is usually better than the performance of tracking that uses only one kind of measurement.

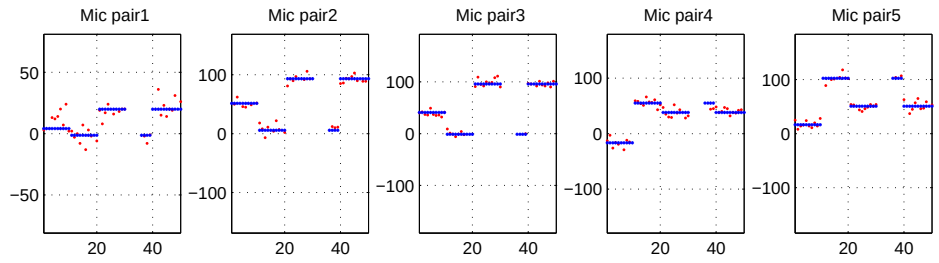
Table 6.22: The performance of joint tracking when only one person speaks among multiple people.

Error condition		Joint tracking (x,y,z)[cm]
$\lambda_{m,a} = 0.0$, $\sigma_a^2 = 0$	Target 1*	(3.6,2.9,2.6)
$\lambda_{m,v} = 0.2$, $\sigma_v^2 = 100$	Target 2	(7.1,5.2,4.5)
$\lambda_{m,a} = 0.0$, $\sigma_a^2 = 0$	Target 1	(7.9,5.8,4.9)
$\lambda_{m,v} = 0.2$, $\sigma_v^2 = 100$	Target 2*	(3.8,2.5,2.2)
$\lambda_{m,a} = 0.2$, $\sigma_a^2 = 50$	Target 1*	(5.6,4.6,5.7)
$\lambda_{m,v} = 0.2$, $\sigma_v^2 = 100$	Target 2	(7.9,5.8,4.4)
$\lambda_{m,a} = 0.2$, $\sigma_a^2 = 50$	Target 1	(8.4,6.1,4.9)
$\lambda_{m,v} = 0.2$, $\sigma_v^2 = 100$	Target 2*	(5.1,3.9,4.5)

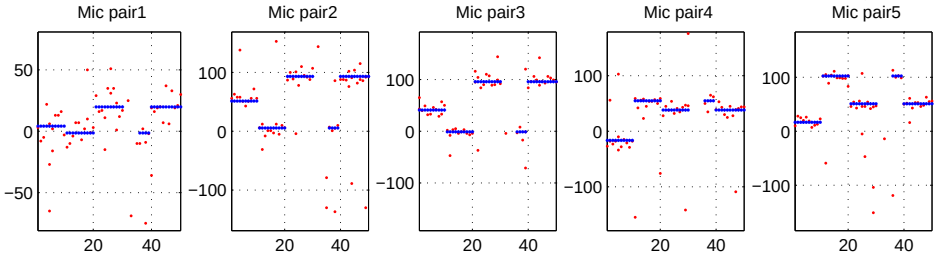
6.4.4 Speaker Activity Detection

The simulation for speaker activity detection is carried out with three people in fixed locations, as shown in Figure 5.16(a). The current speaker switches according to Figure 5.16(b). From time 31 through 35 in Figure 5.16(b), there is silence. To evaluate the effect of missing data or falsely detected data of acoustic measurements on speaker activity detection, a missing data rate of $\lambda_{m,a} = 0.2$ and a falsely detected data rate of $\lambda_{f,a} = 0.2$ are used. The visual measurements have additive Gaussian noise with $\sigma_a^2 = 50$ and $\sigma_v^2 = 100$ and the missing data rate of $\lambda_{m,v} = 0.2$. Sample measurements are shown in Figure 6.4.

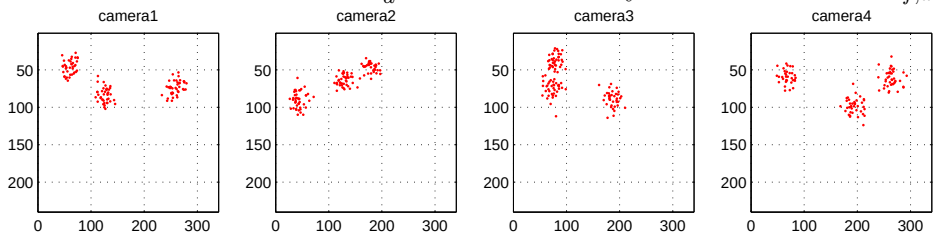
The results of speaker activity detection are shown in Figure 6.5. Without missing data or falsely detected data in the measurements, the speaker activity detection is perfectly accurate. With the missing data, the detection of the current speaker is missed at times $20, 23, 26, 37, 38, 42, 43, 44$, as shown in Figure (b). This is reasonable because there are smaller numbers of measurements when the measurements are lost,



(a) Acoustic measurements with $\sigma_a^2 = 100$ and missing data of $\lambda_{m,a} = 0.2$



(b) Acoustic measurements with $\sigma_a^2 = 100$ and falsely detected data of $\lambda_{f,a} = 0.2$



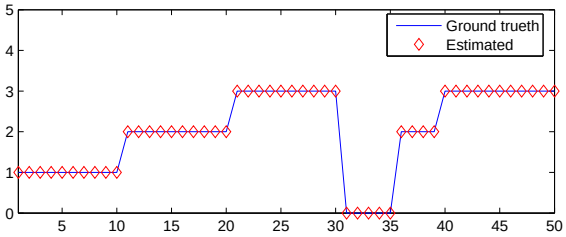
(c) Visual measurements with $\sigma_a^2 = 100$ and missing data of $\lambda_{m,v} = 0.2$

Figure 6.4: Measurements of three people at fixed positions. The blue dots are true measurements, and the red show actual measurements.

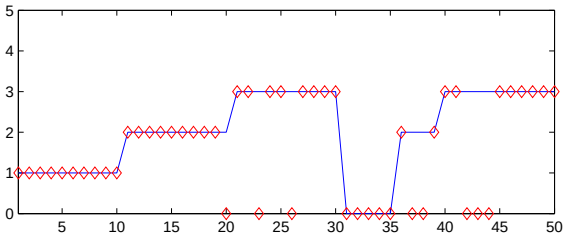
and this results in an increased value of β_{0k}. False alarms, on the other hand, do not affect the result as much as the missing data, but they cause incorrect speaker activity detection at times 5 and 9. This is reasonable because the falsely detected data around the other objects can increase the possibility that a person might be detected as a speaker. In this simulation with 20 repetitions, there are around 20% misses of the true speaker in a missing data environment with $\lambda_{m,a} = 0.2$ and 5% false detections at a falsely detected data rate of $\lambda_{f,a} = 0.2$.

6.5 Discussion

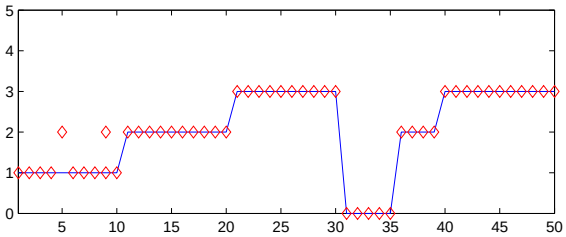
This chapter implemented joint target tracking with multiple cameras and multiple microphones. Using different sensors can improve the entire performance when one



(a) With $\sigma_a^2 = 100$ without any missing data and falsely detected data



(b) With $\sigma_a^2 = 100$ and missing data of $\lambda_{a,m} = 0.2$



(c) With $\sigma_a^2 = 100$ and falsely detected data of $\lambda_{a,f} = 0.2$

Figure 6.5: The results for speaker switching with different error conditions.

kind of measurement is highly contaminated. Even when both measurements are severely contaminated, the chance of having better performance in joint tracking is higher than it is when using only one kind of sensors because the noise sources for the different kinds of sensors are uncorrelated.

CHAPTER VII

REAL-TIME IMPLEMENTATION

7.1 Overview

This chapter addresses the real-time implementation of joint acoustic-visual tracking system using a PC, four cameras, and six microphones. This chapter is composed of two parts: implementation of a real system and real-time processing. The part dealing with the implementation of a real system describes a PC-based tracking system with four cameras, six microphones, and interfaces between these cameras/microphones and the PC. The section on the implementation of real-time processing describes those aspects that make the system work in real time by reducing both processing time and computational complexity.

7.2 Implementation of a Real System

In a room that is approximately 6(W) x 4(L) x 3(H) m^3, four cameras are installed near the corners of the room, as shown in Figure 7.1. Six microphones are located at the positions shown in Table 3.4.

In the past, in order to implement a real tracking system using multiple cameras,

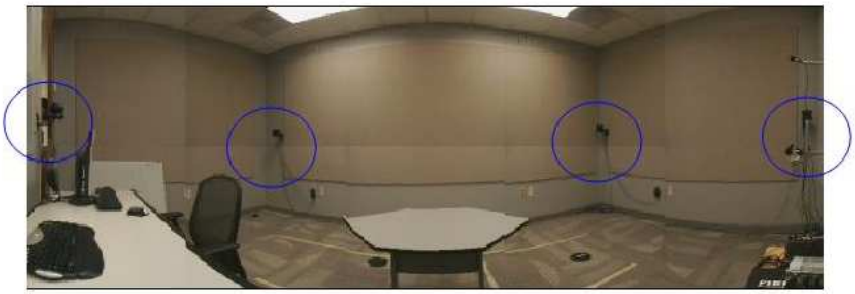


Figure 7.1: A conference room with four cameras.

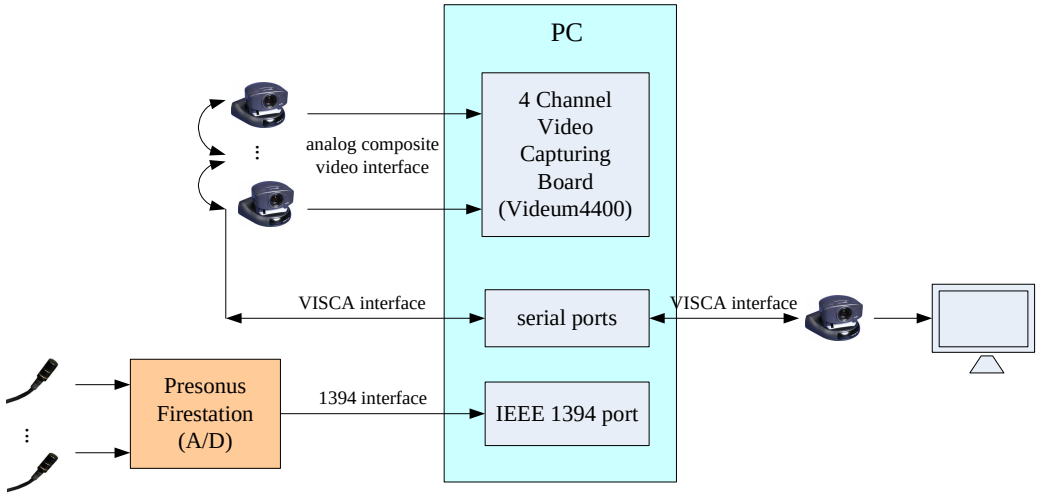


Figure 7.2: The system configuration.

only one camera was connected to a computer because the processing power of the computer was not very high. Each computer processed its camera's images and extracted the visual features, which were then collected in a central computer, which was assigned for tracking. The computer assigned for the tracking processed the transmitted visual features to extract the final interest of the targets [58]. However, this use of multiple computers, as well as the necessary interfaces between computers, required a great deal of time and effort to function properly.

This thesis implements a single PC-based tracking system. The prototype of this system came from Huang [31], who implemented an acoustic-source localization system using TDOAs based on a PC. This system worked quite well.

In the system in this research, all cameras and microphones are connected into the PC, as shown in Figure 7.2. One Videum4400 card, which has four analog video inputs, is installed in the PC to capture the four synchronized camera images. Four SONY EVI-D30 cameras are connected into the Videum4400 card. The control port of one camera is connected into a serial port of the PC, and the control ports of the other cameras are connected in a daisy-chain. All the cameras are controlled using the SONY VISCA control interface from the PC.

Six microphones are connected into an audio A/D converter, which captures acoustic signals at 44.1KHz or 48KHz sampling rates synchronously and sends them to the PC via a 1394 interface. A PortAudio open source library [57] is used to drive the interface in order to send audio streams to the PC through the 1394 interface.

The user interface of the system is shown in Figure 7.2. The whole user interface is divided into two parts of input monitoring and output monitoring. The left side of the user interface is primarily designed for monitoring input signal from cameras and microphones. It also gives intermediate status of visual feature detection to know how well it can detect visual features. The three windows of the figure show the results of the Viola-Jones detector and corner detection. The right top side of the output monitoring part is for showing the entire system status and rough details of each sub-system. The figure says that the entire system works in off-line mode using avi files, and Camera 1 and Camera 2 processed both Viola and corner detection and detected objects, but Camera 3 did not detect any object. The middle black box shows the object followed by extra camera, which is designed to steer to one object. It does not show any image in the figure because the system in Figure 7.2 works in off-line mode. The bottom part of it shows accumulated x and y locations of the tracked objects since the system was started. The figure of the user interface was taken when it processes 46^{th} frame after the system began.

The tracking performance is demonstrated with the use of one extra camera, whose control port is connected to a control port on the PC, and the camera's video port is then connected to a monitor. During the demonstration, the PC steers the camera to follow a one person or a current speaker.

As an intermediate step, this system saves the video streams from the four cameras in AVI video format and the wave data from the six microphones in PCM file format. These data are read by Matlab, which uses the data for algorithm development.

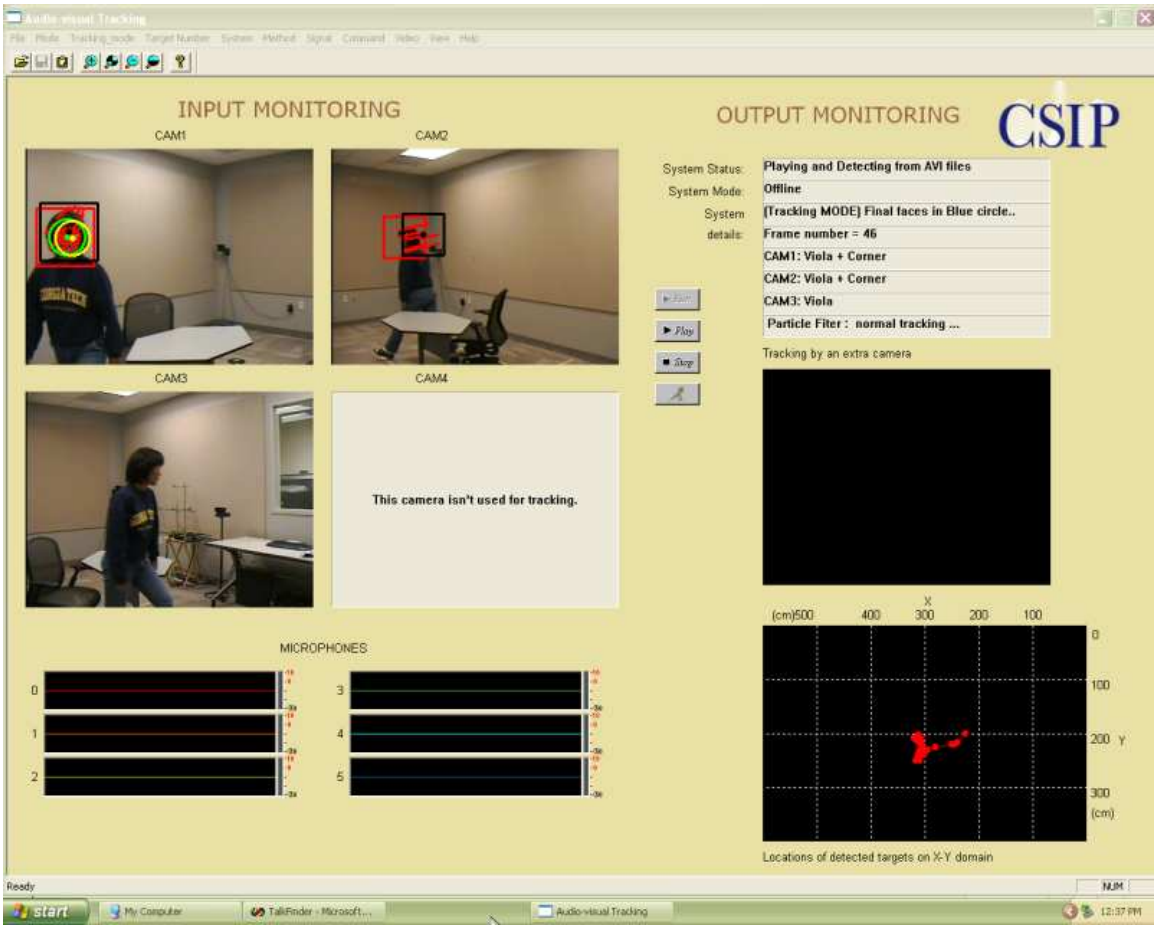


Figure 7.3: The user interface for the system.

7.2.1 Synchronization between Cameras and Microphones

The acoustic capturing board and the multiple video capturing board do not work synchronously here. To implement joint tracking with acoustic data and visual data together, synchronization of both data types is necessary. Since both boards do not send a time code in their streams for synchronization, the synchronization has to be done manually. The initial, synchronized starting point for both audio and video streams is achieved by delaying or advancing the audio steam relative to the video frame. This is done by changing the size of the input buffer of the acoustic stream.

Once the two kinds of data have been captured by each board and processed in the feature-detection programs, the frame rate for capturing the video drops automatically if the processing power cannot process all the frames captured. However, even in this situation when the video frame rate is decreased, the audio interface continuously saves the audio stream into the input buffer and potentially causes the audio buffer to overflow. To solve this overflow problem, this research deletes a certain number of audio frames, as described in Figure 7.4.

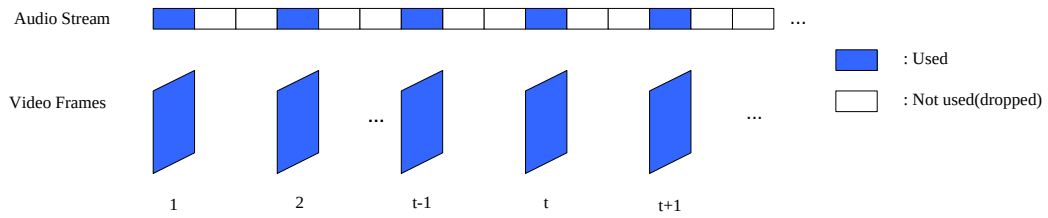


Figure 7.4: The synchronization between audio steams and video frames.

A rectangular block in the audio stream in the figure represents one frame to be used for TDOA estimation. If the frame rate for the video decreases, a pre-calculated number of audio frames, which are white rectangles in the figure, are deleted, as shown in the figure. To solve this overflow problem, the audio steam can be down-sampled, but a high rate of down-sampling distorts the input audio signals and prohibits accurate estimation of TDOAs.

7.3 Implementation of a Real Time Processing

Since multiple cameras and microphones are connected into just one computer, the entire computational complexity is high. To make the system track targets in real time, several things should be considered.

7.3.1 Low Resolution and Low Frame rate

Since four cameras are connected to only one PC, 620 x 480 image resolution cannot be used because such a high resolution requires too much computation and data transfer between the video capturing card and the CPU. Therefore, this system uses 320 x 240 image resolution. Since the images are color images, the data rate for each frame is 320 x 240 x 3. Even with the limited image size, the system still requires considerable computation to process all four camera images. Thus, the frame rate decreases from less than 5 frames/sec, where 5 frames/sec is the frame rate when camera images are saved to avi files in the PC. Since this system aims to track people in a conference room, and their movement is not fast but is irregular, a low frame rate is acceptable. When the system cannot process all the data, it is the frame rate that decreases even further, not the image resolution.

7.3.2 System Operation of Initialization and Tracking

The system operates in the two modes of initialization and tracking. Initialization can work in non-real time, but tracking needs to work in real time, even though the actual frame rate might be very low. During initialization, the number of targets and the initial positions of the targets are estimated. This follows the procedure in Table 4.10. Since the tracking performance greatly depends on the performance of the initialization, accurate initialization is required. This accuracy is not easy to achieve in real time. Therefore, the initialization does not aim to work in real time. For visual-feature detection, the three Viola-Jones detectors are used to find

their assigned features in the entire image. The detection using the three Viola-Jones detectors is performed over several frames, and the results are overlapped for greater accuracy. In speaker tracking, the initialization starts when there is available acoustic data, and is repeated for several frames to get more accurate results. For joint audio-visual tracking, the initialization uses only visual data.

After initialization, tracking begins. During tracking, the number of targets is assumed to be fixed. Visual-feature detection uses motion vectors by using corner detection/matching algorithms. The system uses a small variance in the state update model. The tracking uses only a small number of particles, fewer than 500 for 3D searches. The proposal function uses a state update model with a small variance.

7.3.3 Vector Processing

For real-time processing, the parallelization of vector operations is necessary. Since most of the data take the form of matrices or vectors, the parallelization and vectorizing operations can save processing time. A user or programmer can optimize operations, but another good way to handle vectorization is to use a commercial library. The current system does not use any commercial library to parallelize vector operations, so its processing speed is quite slow. A good library for this vectorization is Intel Integrated Performance Primitives (IPP) because the library was optimized for Intel processors. If incorporated this would likely improve system performance.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 Summary of the Thesis Contributions

Applications such as video conferencing, visual surveillance, and scene analysis require tracking the locations of subjects in common. These applications also use acoustic or visual data to track subjects. This thesis proposed the tracking of three-dimensional (3D) locations of people and the current speaker/speakers in a real conference room environment. The research also sought to implement tracking algorithms in a real system. To achieve these objectives, the research proposed joint audio-visual tracking using both acoustic and visual data to complement each other's limitations. The research focused mainly on robust acoustic-feature detection algorithms for multiple speakers, robust face/head detection algorithms for non-rigid objects, a tracking framework to track multiple targets using measurements from multiple microphones and cameras, and data association and sensor fusion algorithms for multiple targets in the particle filter.

The main contribution of the proposed research is to develop a non-linear Bayesian multiple-target tracking system using both microphones and cameras. The proposed algorithms and the procedures for each application can be easily extended or modified to different tracking scenarios such as vehicle tracking, pedestrian tracking, etc.

The research also demonstrated multiple acoustic source tracking with a small number of microphones using TDOAs with reasonable accuracy. Despite the fact that localization using TDOA has been known to detect only a single speaker, this research demonstrated that data association combined with TDOA features was able to track multiple targets. The main contribution here is to extract multiple TDOAs for the

cases where reverberation and a high noise level are present and solve the association of target-to-measurements, false alarms, or missing data in the data association.

The research proposed fast, reliable visual-feature detection using motion vectors using a corner feature. Even though corner detection is used for many applications in computer vision, it has not been used for motion detection for non-rigid objects as is proposed here.

This research applied a particle filter as the main tracking framework to implement non-linear measurement models and to simplify the fusion of different modalities for joint tracking. Using a particle filter for tracking is not new. However, designing an appropriate state space and observation models for a specific tracking scenario is novel and valuable.

This research also implemented and evaluated two data association methods, MC-JPDAF and DA-IPPF, which were basically proposed by Vermaark, in different noise environments and in different tracking scenarios. MC-JPDAF was proved to be suitable enough to use for our tracking scenarios. MC-JPDAF worked quite well even in the conditions of a high missing data rate and Gaussian measurement noise. The research improved the data association method using IPPF by proposing an exclusive data association parameter, but MC-JPDAF showed more stable performance than DA-IPPF, especially when measurements for multiple targets were close. The research revealed that the assumption of independent sampling of the data association parameter in IPPF for multiple targets was not reasonable because data association tries to figure out which measurement belongs to which target. This indicates that data association parameters for multiple targets should be dependent.

The research also proposed an initialization method, which is very critical to most tracking algorithms. This initialization method worked quite well in moderate noisy environments in different application scenarios of multiple speaker tracking and multiple people tracking.

The research also proposed a method to estimate speaker activity from the data association without any extra calculation.

Finally, this research implemented the entire tracking algorithms in a real system based on using a single PC. Even though this system cannot work at high frame rates, it demonstrates promising results.

8.2 Future Works

The research points to several topics for future research to improve the performance of the proposed algorithms and to apply the proposed algorithms to different applications.

First, the research can be expanded by using steerable cameras. Since the system knows the positions of targets at the previous frame time, it can steer the cameras to the specific target. Then, the camera calibration matrix can be calculated using only the steering information without recalibration in real time. In this case, even though the angle of view of the camera is limited, since it can be steered to the target, there would be a reduced chance of losing the target. This thesis introduces a method for performing the camera calibration using the steering information of the camera in Appendix A. One possible application using this scenario is to track a single speaker using multiple microphones and one steering camera. First, the initial position of the speaker is calculated by simple acoustic-source localization using multiple microphones. Then, the camera is steered to the initial position from the acoustic-source localization. Next, for more robust tracking, joint tracking is applied using acoustic data and a face extracted from the steered camera, which uses a camera-calibration matrix calculated with the steered information.

The next area of future work would be to use multiple view geometry between multiple cameras to improve the performance and reduce of visual-feature extraction more than we used in Section 3.4. This research only used the epipolar geometry to

determine the robustness of detected features from only a single classifier. However, if we use more information from detected objects in the previous frames, using multiple camera geometry can improve the detection rate of objects. It can also contribute to reduce computation time, which is a big limitation in a real system.

A third area for future research is that the tracking of faces/heads with motion by using corner detection/matching can be improved, especially for the cases in which a person approaches a camera. The window size of a non-rigid object can be controlled by using the location of the object in tracking.

The last suggestion for future research is to use Intel IPP to speed up the processing time in the system. Most operations in acoustic-feature detection, visual-feature detection, and particle filtering use vectors and matrices, but the current programs in the system were not optimized for these vector/matrix operations. If these parts are replaced with a library of Intel IPP, the system could be expected to operate much faster than the current system.

APPENDIX A

CAMERA CALIBRATION

Camera calibration [25] describes the relationship between a 3D position in Cartesian coordinates and a 2D image point. To derive the camera-calibration matrix for a camera, a simple pin-hole model is usually used as a camera model. If one point in 3D Cartesian coordinates is denoted by (x, y, z), the pin-pole model maps this point into a point in a camera image, which is $f(\frac{x}{z}, \frac{y}{z})$, using a focal length f of the camera. By using homogeneous coordinates [], a $2D$ point (x, y) is denoted by $(x, y, 1)$. Likewise, one 3D point, (x, y, z), is denoted by $(x, y, z, 1)$ using the homogeneous coordinates. The pin-hole model using the homogeneous coordinates is also described as

$$\begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{93}$$

Denoting $(x, y, z, 1)^T$ as $\mathbf{X}$ and $(f\frac{x}{z}, f\frac{y}{z}, 1)^T$ as $\mathbf{x}$, Equation (93) simplifies to

$$\mathbf{x} = k \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X} = k\mathbf{P}\mathbf{X} \tag{94}$$

where $\mathbf{P}$ is called a pin-hole camera projection matrix.

However, since the camera center using Equation (94) maps into the center of an image, the origin should be moved into the camera's principal point, (p_x, p_y). Then,

Equation (94) is changed to

$$\mathbf{x} = k \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X} = k \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X} = k\mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X}. \quad (95)$$

K is called the camera-calibration matrix, more specifically the internal camera-calibration matrix, to distinguish it from an external camera-calibration matrix. However, an image in a CCD camera is described by a pixel and cannot be guaranteed to be square pixels. The pixel aspect ratio is denoted as (m_x, my). Therefore, (fm_x, fm_y) is replaced with (α_x, α_y) and the principal point is also defined (u_0, v_0) in terms of pixel dimensions. In addition, the lens distortion, called a skew, s, is also reflected even though it is zero in most cameras. As a result, the final **K**, the internal camera-calibration matrix, is

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The internal camera-calibration matrix is not enough to describe the mapping between a 3D point and a 2D image point because the origin of 3D Cartesian coordinates is different from the origin of the camera and because the direction of 3D Cartesian coordinates can be different from the direction of the camera. These differences in the origin and direction of the coordinate systems are expressed by a rotation and a translation. Denote **X** in Equation (95) by **Xcam** with a reference at the camera center with the same direction of the camera and **X** as a point in any 3D coordinate system. The mapping between **X** and **Xcam** is described using the translation and the rotation as

$$\mathbf{Xcam} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X} = [\mathbf{R}|\mathbf{t}]\mathbf{X}. \quad (96)$$

$\mathbf{R}$ and $\mathbf{C}$ are the rotation matrix and the translation vector between $\mathbf{Xcam}m$ and $\mathbf{X}$. When this relation is applied to Equation (95), the final equation is

$$\mathbf{x} = k\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} = k\mathbf{P}\mathbf{X}. \tag{97}$$

To calibrate a camera, a DLT (Direct Linear Transformation) or Gold Standard algorithm [25] has been widely used. In practice, the most well-known camera-calibration tool came from CALTECH [1], which was also implemented in Intel OpenCV [35]. This technique requires only a few planar patterns with different orientations. When a steering camera is used, camera movement can be modeled as a pan and a tilt as $\mathbf{R_{Pan}}$ and $\mathbf{R_{Tilt}}$. If θ and ϕ are the panning angle and the tilting angle, then $\mathbf{R_{Pan}}$ and $\mathbf{R_{Tilt}}$ are as follows.

$$\mathbf{R_{Tilt}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) & 0 \\ 0 & sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_{Pan}} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now, $\mathbf{R}$ in Equation (96) is multiplied with $\mathbf{R_{Tilt}}\mathbf{R_{Pan}}$, and then the calibration matrix of a steering camera is

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] = \mathbf{K}[\mathbf{R}_{Tilt}\mathbf{R}_{Pan}\mathbf{R}|\mathbf{t}] \tag{98}$$

where $\mathbf{t}$ is the transition vector between the position of the camera and the reference point for 3D Cartesian coordinates.

APPENDIX B

EPIPOLAR GEOMETRY

The epipolar geometry [25] is the intrinsic geometry between two views, and a fundamental matrix encapsulates this epipolar geometry in an uncalibrated case; an essential matrix encapsulates this epipolar geometry in a calibrated case. Here, without distinguishing between the fundamental matrix and the essential matrix, this section calls it a fundamental matrix, in general. For more than two views, a trifocal tensor [25] plays the same role as the fundamental matrix. This trifocal tensor can be extended to more than three views. Both the epipolar geometry and the trifocal tensor are independent of the scene structure but dependent on all the projective geometry between views. Figure B.1 shows a mapping $\mathbf{X}$ in 3D Cartesian coordinates into $\mathbf{u}$ and $\mathbf{u}^{'}$ in two different views. $\mathbf{e}$ and $\mathbf{e}^{'}$ are epipoles, and $\mathbf{c}$ and $\mathbf{c}^{'}$ are camera centers. The point $\mathbf{X}$ maps into $\mathbf{u}$ in one view and $\mathbf{u}^{'}$ in the other view. Although $\mathbf{u}$ is assumed to be known, $\mathbf{u}^{'}$ cannot be estimated from $\mathbf{u}$. However, $\mathbf{u}^{'}$ is on the line $\mathbf{l}^{'}$ if the fundamental matrix is known. This relationship of a point in one camera into a line in another camera can be described by the fundamental matrix, $\mathbf{F}$, which is a 3x3 matrix by

$$\mathbf{l}' = \mathbf{F}\mathbf{u}. \tag{99}$$

Since point $\mathbf{u}^{'}$ is on the $\mathbf{l}^{'}$, the inner product of $\mathbf{u}^{'T}$ and $\mathbf{l}^{'}$ is zero, such as $\mathbf{u}^{'T}\mathbf{l}^{'} = 0$. When $\mathbf{l}^{'}$ is replaced with Equation (99), the final relationship between $\mathbf{u}$ and $\mathbf{u}^{'}$ using $\mathbf{F}$ is

$$\mathbf{u}^{'T}\mathbf{F}\mathbf{u} = 0. \tag{100}$$

Equation (100) should always be satisfied for any pair of the corresponding points

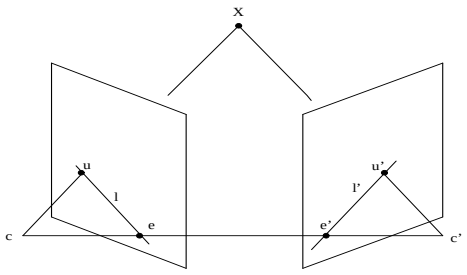


Figure B.1: The correspondence between a point in 3D and a point in an image.

of two different views. Calculating $\mathbf{F}$ between two different views can be done using either at least seven correspondences or the camera-calibration matrices of the two views. Since all cameras are calibrated in this research, $\mathbf{F}$ is calculated using the camera-calibration matrices. Among several ways to calculate $\mathbf{F}$ using two calibration matrices, a 4x4 determinant using rows of the camera matrices is used [25].

As in Equation (98), assume there are two cameras whose camera calibration matrices are $\mathbf{P}$ and $\mathbf{P'}$; then, the projecting points of $\mathbf{X}$ onto two views are $\mathbf{u} = k\mathbf{P}\mathbf{X}$ and $\mathbf{u}^{'} = k\mathbf{P}^{'}\mathbf{X}$. These equations can be written as

$$\begin{bmatrix} \mathbf{P} & \mathbf{u} & 0 \\ \mathbf{P}^{'} & 0 & \mathbf{u}^{'} \end{bmatrix} \quad \begin{bmatrix} \mathbf{X} \\ -1/k \\ -1/k^{'} \end{bmatrix} = \mathbf{0}. \tag{101}$$

By denoting $\mathbf{P}$ as $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^T$ with their row vectors, Equation (101) can be written as

$$\begin{bmatrix} \mathbf{p}_1 & u_1 & 0 \\ \mathbf{p}_2 & u_2 & 0 \\ \mathbf{p}_3 & 1 & 0 \\ \mathbf{p}_1^{'} & u_1^{'} & 0 \\ \mathbf{p}_2^{'} & u_2^{'} & 0 \\ \mathbf{p}_3^{'} & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} \mathbf{X} \\ -1/k \\ -1/k^{'} \end{bmatrix} = \mathbf{0}. \tag{102}$$

If $(\mathbf{X}, -1/k, -1/k^{'})^T$ has a non-zero solution, the determinant of the matrix in

Equation (102) should be zero. Therefore, the determinant of Equation (102) is

$$u_1\left(u_1^{'}\begin{vmatrix}\mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_2^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}-u_2^{'}\begin{vmatrix}\mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_1^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}+\begin{vmatrix}\mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_1^{'}\\ \mathbf{p}_2^{'}\end{vmatrix}\right)-u_2\left(u_1^{'}\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_3\\ \mathbf{p}_2^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}-u_2^{'}\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_3\\ \mathbf{p}_1^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}+\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_3\\ \mathbf{p}_1^{'}\\ \mathbf{p}_2^{'}\end{vmatrix}\right)+\left(u_1^{'}\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_2^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}-u_2^{'}\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_1^{'}\\ \mathbf{p}_3^{'}\end{vmatrix}+\begin{vmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_1^{'}\\ \mathbf{p}_2^{'}\end{vmatrix}\right)=0. \tag{103}$$

After identifying each term in Equation (103) with the corresponding terms from Equation (100), an element of the fundamental matrix, f_{ij}, is described by the determinant of two camera-calibration matrices as

$$f_{ij}=(-1)^{i+j}det\begin{bmatrix}\sim\mathbf{P}_j\\ \sim\mathbf{P}_i^{'}\end{bmatrix} \tag{104}$$

where $\sim\mathbf{P}_i$ is $\mathbf{P}$ without row i.

After a point is obtained from one camera and multiplied with the fundamental matrix of another camera by using Equation (104).

REFERENCES

[1] "Camera calibration tool box and browsing," `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[2] Aarabi, P. and Zaky, S., "Integrated vision and sound localization," *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, vol. 2, July 2000.

[3] Alarm, M. and McClellan, K. H., "Near field imaging of subsurface targets using active arrays and elastic waves," pp. 216– 220, Aug. 2004.

[4] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T., "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," in *Signal Processing, IEEE Transactions on*, vol. 50, pp. 174 – 188, Feb. 2002.

[5] B. Wu, X. Song, V. S. and Nevatia, R., "Evaluation of usc human tracking system for surveillance videos," *In CLEAR Evaluation Campaign and Workshop*, 2006.

[6] Bar-Shalom, Y. and Fortman, T. E., *Tracking and Data Association*. Mathematics in Science and Engineering, 1988.

[7] Bar-Shalom, Y. and Fortmann, T. E., *Tracking and Data Association*, vol. 179. Academic Press Professional, Inc., 1988.

[8] Cevher, V. and McClellan, K. H., "Tracking of multiple wideband targets using passive sensor arrays and particle filters," pp. 72– 77, 2002.

[9] Checka, N., Wilson, K., Rangarajan, V., and Darrell, T., "A probabilistic framework for multi-modal multiperson tracking," in *2003 Conference on Computer Vision and Pattern Recognition Workshop*, p. 100, 2003.

[10] Chen, J. C., Hudson, R. E., and Yao, K. Y., "Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," *Signal Processing, IEEE Transactions on*, vol. 50, no. 8, pp. 1843–1854, Aug. 2002.

[11] Chen, J., Yao, K., and Hudson, R., "Source localization and beamforming," *Signal Processing Magazine, IEEE*, vol. 19, pp. 30–39, March 2002.

[12] Chen, Z., "Bayesian filtering: From kalman filters to particle filters, and beyond," in *Technical report,*, vol. 7, pp. 723–733, May 2007.

[13] CHIL, "Computers in the human interaction loop." `http://chil.server.de/servlet/is/101/`.

[14] Comaniciu, D. and Meer, P., "Robust analysis of feature spaces: color image segmentation," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 750 – 755, June 1997.

[15] Comaniciu, D. and Meer, P., "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 603 – 619, May 2002.

[16] Comaniciu, D., Ramesh, V., and Meer, P., "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, pp. 142–149, 2000.

[17] Doucet, A., Freitas, N., and Gordon, N., *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[18] Freund, Y. and Schapire, R. E., "Experiments with a new boosting algorithm," *International Conference on Machine Learning*, pp. 148–156, 1996.

[19] Gatech, "The aware home," `http://www.awarehome.gatech.edu/index.html`.

[20] Gordon, N., "A hybrid bootstrap filter for target tracking in clutter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 33, pp. 353 – 358, Jan 1997.

[21] Gordon, N., "A hybrid bootstrap filter for target tracking in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, pp. 353–358, 1997.

[22] Hara, I., Asano, F., Asoh, H., Ogata, J., Ichimura, N., Kawai, Y., Kanehiro, F., Hirukawa, H., and Yamamoto, K., "Robust speech interface based on audio and video information fusion for humaniod hrp-2," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2404 – 2410, Oct. 2004.

[23] Haritaoglu, I., Harwood, D., and Davis, L. S., "W4: Who? when? where? what? a real time system for detecting and tracking people," *In Proceedings of the Third Face and Gesture Recognition Conference*, pp. 222–227, 1998.

[24] Harris, C. and Stephens, M., "A combined corner and edge detector," in *In ALVEY Vision Conference*, pp. 147–151, 1988.

[25] Hartley, R. and Zisserman, A., *Multiple view geometry in computer vision*. Cambridge, 2003.

[26] Herbordt, W., *Sound Capture for Human/Machine Interfaces*. Springer-Verlag, 2005.

[27] Hogg, R. V., McKean, J. W., and Craig, A. T., *Introduction to Mathematical Statistics, sixth edition*. Prentice Hall, 2005.

[28] HoseinNezhad, R., Moshiri, B., and Asharig, M. R., "P2-40:sensor fusion for ultrasonic and laser array in mobile robots:a comparative study of fuzzy, dempster and bayesian approaches," in *Sensors, 2002. Proceedings of IEEE*, vol. 2, pp. 1682– 1689, 2002.

[29] H.Seitner, F. and Lovell, B. C., "Fast pedestrian tracking based on spatial features and colour," in *Computer vision winter workshop 2006*, Feb. 2006.

[30] Hsu, R., Abdel-Mottaleb, M., and Jain, A., "Face detection in color images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 696–706, May 2002.

[31] Huang, Y., *Real-time acoustic source localization with passive microphone arrays*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2001.

[32] Huang, Y. and Barkat, M., "Near-field multiple source localization by passive sensor array," *Antennas and Propagation, IEEE Transactions on*, vol. 39, pp. 968 – 975, July 1991.

[33] Huang, Y., Elko, G. W., and Mersereau, R. M., "Real-time passive source localization: a practical linear-correction least-square approach," in *Speech and Audio Processing, IEEE Transactions on*, vol. 9, pp. 943–956, 2001.

[34] II, M. L., Chong, C., Kadar, I., Alford, M., Vannicola, V., and Thomopoulos, S., "Distributed fusion architectures and algorithms for target tracking," *Proceedings of the IEEE*, vol. 85, pp. 95 – 107, Jan. 1997.

[35] INTEL, "Open source computer vision library." `http://www.intel.com/research/mrl/research/opencv/`.

[36] Isard, M. and MacCormick, J., "Bramble: A bayesian multiple-blob tracker," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 34–41, 2001.

[37] Ivanov, Y. A., Bobick, A. F., and Liu, J., "Fast lighting independent background subtraction," *Visual Surveillance, 1998. Proceedings., 1998 IEEE Workshop on*, pp. 49 – 55, Jan. 1998.

[38] Johnson, D. and Dudgeon, D., *Array Signal Processing: Concepts and Techniques*. Prentice Hall, 1993.

[39] Knap, C. and Carter, G., "The generalized correlation method for estimation of time delay," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 24, pp. 353–358, 1976.

[40] Krim, H. and Krim, M. V., "Two decades of array signal processing research: the parametric approach," *Signal Processing Magazine, IEEE*, vol. 13, pp. 67–94, Jul 1996.

[41] Kruppa, H., Santana, M. C., and Schiele, B., "Fast and robust face finding via local context," in *in Proc. VSPETS,*, pp. 157–164, 2003.

[42] Kumar, M., Garg, D. P., and Zachery, R. A., "A method for judicious fusion of inconsistent multiple sensor data," in *Sensors Journal, IEEE*, vol. 7, pp. 723–733, May 2007.

[43] Lehman, E. A., "Image-source method." `http://www.watri.org.au/~ericl/`.

[44] Lehmann, E. L. and Romano, J. P., *Testing Statistical Hypothesis, third edition*. Springer, 2005.

[45] Lepetit, V., Shahrokni, A., and Fua, P., "Particle phd filtering for multi-target visual tracking," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–281– I–288, June 2003.

[46] Lienhart, R. and Maydt, J., "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. 900–903, 2002.

[47] MacCormick, J. and Isard, M., "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *ECCV*, pp. 3–19, 2000.

[48] Marczak, A. L. C., Johnson, M. J., and Messom, C. H., "Real-time computation of haar-like features at generic angles for detection algorithm," *Res. Let. Inf. Math. Sci*, vol. 9, pp. 98–111, 2006.

[49] Mathes, T. and Piater, J., "Robust non-rigid object tracking using point distribution manifolds," in *28th Annual Symposium of the German Association for Pattern Recognition*, 2006.

[50] Merwe, E. W. R. V. D., "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000, AS-SPCC*, pp. 153 – 158, 2000.

[51] Mikolajczyk, K. and Schmid, C., "Scale and affine invariant interest point detectors," in *Int. J. Comput. Vision*, vol. 60, pp. 63–86, Oct. 2004.

[52] Mittal, A. and Davis, L. S., "M2tracker: a multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo," in *Proc. of European Conf. on Computer Vision*, pp. 18–33, 2002.

[53] Orton, M. and Fitzgerald, W., "A bayesian approach to tracking multiple targets using sensor arrays and particle filters," *Signal Processing, IEEE Transactions on*, vol. 50, pp. 216–223, Feb 2002.

[54] Perez, P., Vermaak, J., and Blake, A., "Data fusion for visual tracking with particles," *Proceedings of the IEEE*, vol. 92, pp. 495–513, Mar 2004.

[55] Phung, P., Bouzerdoum, A., and Chai, D., "A novel skin color model in ycbcr color space and its application to human face detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, 2002.

[56] Rasmussen, C. and Hager, G., "Probabilistic data association methods for tracking complex visual objects," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 560 – 576, Jun 2001.

[57] Ross Bencina, e., "Portauio:portable cross-platform audio api." `http://www.portaudio.com/`.

[58] Ruddarraju, R. and Essa, I. A., "Fast multiple camera head pose tracking," in *In Proceedings, Vision Interface 2003*, 2003.

[59] Sattar, J. and Dudek, G., "the performance of color tracking algorithms for underwater robots under varying lighting and visibility," in *In International Conference on Robotics and Automation, ICRA 2006*, 2006.

[60] Schmid, C., Mohr, R., and Bauckhage, C., "Evaluation of interest point detectors," vol. 37, pp. 151–172, 2000.

[61] Simon, G., Fitzgibbon, A., and Zisserman, A., "Markerless tracking using planar structures in the scene," in *IAugmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pp. 120–128, 2000.

[62] Smith, J. O. and Abel, J. S., "Closed-form least-squre source location estimation from range-difference measurements," in *Acoustics, Speech, and Signal Processing, IEEE Transactions on*, vol. 35, pp. 1661 – 1669, Dec. 1987.

[63] Stauffer, C. and Grimson, W., "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, p. 2246, 1999.

[64] Stoica, P. and Nehorai, A., "Music, maximum likelihood, and cramer-rao bound," *Acoustics, Speech, and Signal Processing, IEEE Transactions on*, vol. 37, no. 5, pp. 720–741, May 1989.

[65] Tian, Y., Lu, M., and Hampapur, A., "Robust and efficient foreground analysis for real-time video surveillance," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 1182 – 1187, June 2005.

[66] Tiehan, L., Ozer, B., and Wolf, W., "A real-time background subtraction method with camera motion compensation," *Multimedia and Expo, 2004 IEEE International Conference on*, vol. 1, pp. 331 – 334, 2004.

[67] Vermaak, J., Gangnet, M., Blake, A., and Perez, P., "Sequential monte carlo fusion of sound and vision for speaker tracking," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, pp. 741 – 746, July 2001.

[68] Vermaak, J., Godsill, S. J., and Perez, P., "Monte carlo filtering for multi-target tracking and data association," in *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 41, pp. 309 – 332, Jan. 2005.

[69] Viola, P. and Jones, M., "Robust real-time face detection," *International Journal of Computer Vision, 2002*, 2002,.

[70] Viola, P., Jones, M., and Snow, D., "Detecting pedestrians using patterns of motion and appearance," Tech. Rep. TR-2003-90, 2003.

[71] Ward, D. B., Lehman, E. A., and Willianson, R. C., "Particle filter algorithm for tracking an acoustic source in a reverberant environment," *Speech and Audio Processing, IEEE Transactions on*, vol. 11, pp. 826–836, Nov. 2003.

[72] Wu, B., AI, H., Huang, C., and Lao, S., "Fast rotation invariant multi-view face detection based on real adaboost," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FG'04)*, p. 79, 2004.

[73] Wu, H., Fukumoto, T., Chen, Q., and Yachida, M., "Active face observation system," vol. 3, pp. 441–445, 1996.

[74] Yang, M. H., Kriegman, D. J., and Ahuja, N., "Detecting faces in images:a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 1.

[75] Zhang, Z., Li, M., Li, S. Z., and Zhang, H., "Multi-view face detection with floatboost," in *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, pp. 184 – 188, 2002.

[76] Zhou, Y., Yip, P., and Leung, H., "Tracking the direction-of-arrival of multiple moving targets by passive arrays: Algorithm," *Signal Processing, IEEE Transactions on*, vol. 47, no. 10, pp. 2655–2666, Oct. 1999.

VITA

Yeongseon Lee finished her bachelor and master degrees in Electrical Engineering from Kyungpook national university. Then she worked at Image processing and communication lab in Electronics and Telecommunication Research Institute (ETRI). She came to US with her daughter after her husband started to study in Georgia Tech. Then She also started to study in ECE as a part-student in 2001 and turned to study as a full-time student in 2003 with Dr. Mersereau.